The R Journal

Volume 14/1, March 2022

A peer-reviewed, open-access publication of the
R Foundation for Statistical Computing

Contents

Editorial 4

Contributed Research Articles

A Computational Analysis of the Dynamics of R Style Based on 108 Million Lines of Code from All CRAN Packages in the Past 21 Years 6

rmonad: pipelines you can compute on 22

A Software Tool For Sparse Estimation Of A General Class Of High-dimensional GLMs 34

bayesanova: An R package for Bayesian Inference in the Analysis of Variance via Markov Chain Monte Carlo in Gaussian Mixture Models 54

tvReg: Time-varying Coefficients in Multi-Equation Regression in R 79

RKHSMetaMod: An R Package to Estimate the Hoeffding Decomposition of a Complex Model by Solving RKHS Ridge Group Sparse Optimization Problem 101

Measuring the Extent and Patterns of Urban Shrinkage for Small Towns Using R . . . 123

blindrecalc - An R Package for Blinded Sample Size Recalculation 137

Revisiting Historical Bar Graphics on Epidemics in the Era of R ggplot2 146

spherepc: An R Package for Dimension Reduction on a Sphere 167

The smoots Package in R for Semiparametric Modeling of Trend Stationary Time Series182

cpsurvsim: An R Package for Simulating Data from Change-Point Hazard Distributions 196

starvars: An R Package for Analysing Nonlinearities in Multivariate Time Series . . . 208

fairmodels: a Flexible Tool for Bias Detection, Visualization, and Mitigation in Binary Classification Models 227

Palmer Archipelago Penguins Data in the palmerpenguins R Package - An Alternative to Anderson's Irises 244

Advancing Reproducible Research by Publishing R Markdown Notebooks as Interactive Sandboxes Using the learnr Package 255

Power and Sample Size for Longitudinal Models in R – The longpower Package and Shiny App 264

PSweight: An R Package for Propensity Score Weighting Analysis 282

RFpredInterval: An R Package for Prediction Intervals with Random Forests and Boosted Forests 300

etrm: Energy Trading and Risk Management in R 320
fcaR, Formal Concept Analysis with R 341
FMM: An R Package for Modeling Rhythmic Patterns in Oscillatory Systems. 361

News and Notes

Changes on CRAN 380
R Foundation News 382

Prospective authors will find detailed and up-to-date submission instructions on the Journal's homepage.

R Journal Homepage:
http://journal.r-project.org/

Email of editors and editorial board:
r-journal@R-project.org

The R Journal is indexed/abstracted by EBSCO, DOAJ, Thomson Reuters.

Editorial

by Catherine Hurley

On behalf of the editorial board, I am pleased to present Volume 14 Issue 1 of the R Journal. This issue heralds a switch from two issues per year to four and is my first as Editor-in-Chief. The change to four issues per year is in response to the increase in published articles in recent years. As articles will appear more speedily in a published issue, we will no longer list pdfs for Accepted articles on The R Journal website.

First, some news about the journal board. Dianne Cook has stepped down as Editor-in-Chief but continues as an Executive Editor. In her time as EIC she provided excellent leadership and brought in many advances, most notably the change to the new modern journal format. One new Associate Editor, Simone Blomberg, has recently joined the team. We have a new, slimmed-down Editorial advisory board consisting of Henrik Bengtssen, Gabriela de Quiroz, Michael Kane and Rececca Killick. The board will provide continuity across changes in the editorial board, offering advice and acting as an independent body to handle issues of academic integrity.

Behind the scenes, several people are assisting with the journal operations and the new developments. Mitchell O'Hara-Wild continues to work on infrastructure, and H. Sherry Zhang continues to develop the **rjtools** package. In addition, articles in this issue have been carefully copy edited by Hannah Comiskey.

1 In this issue

News from the CRAN and the R Foundation are included in this issue.

This issue features 22 contributed research articles the majority of which relate to R packages for modelling tasks. All packages are available on CRAN. Topics covered are:

- **Temporal and longitudinal methods**
  - **cpsurvsim**: An R Package for Simulating Data from Change-Point Hazard Distributions
  - The **smoots** Package in R for Semiparametric Modeling of Trend Stationary Time Series
  - **starvars**: An R Package for Analysing Nonlinearities in Multivariate Time Series
  - **FMM**: An R package for modeling rhythmic patterns on oscillatory systems
  - **tvReg**: Time-varying Coefficients in Multi-Equation Regression in R
  - Power and Sample Size for Longitudinal Models in R - The **longpower** Package and Shiny App
- **Estimation and inference**
  - **dglars** A Software Tool For Sparse Estimation Of A General Class Of High-dimensional GLMs
  - **bayesanova**: An R package for Bayesian inference in the analysis of variance via Markov Chain Monte Carlo in Gaussian mixture models
  - **RKHSMetaMod**: An R package to estimate the Hoeffding decomposition of a complex model by solving RKHS ridge group sparse optimization problem
  - **PSweight**: An R Package for PropensityScore Weighting Analysis
- **Machine learning**
  - **RFpredInterval**: An R Package for Prediction Intervals with Random Forests and Boosted Forests
  - **fairmodels**: A Flexible Tool For Bias Detection, Visualization, And Mitigation Graphics and Visualisation, Machine Learning & Statistical Learning

  - **spherepc**: An R Package for Dimension Reduction on a Sphere

- **Other topics**
  - **blindrecalc**: An R Package for Blinded Sample Size Recalculation
  - **rmonad**: Pipelines you can compute on
  - **etrm**: Energy Trading and Risk Management in R
  - **fcaR**, Formal Concept Analysis with R
  - Advancing reproducible research by publishing R markdown notebooks as interactive sandboxes using the `learnr` package

- **Applications**
  - Palmer Archipelago Penguins Data in the **palmerpenguins** R Package - An Alternative to Anderson's Irises
  - A Computational Analysis of the Dynamics of R Style Based on 94 Million Lines of Code from All CRAN Packages in the Past 20 Years
  - Measuring the Extent and Patterns of Urban Shrinkage for Small Towns Using R
  - Revisiting Historical Bar Graphics on Epidemics in the Era of R ggplot2

Catherine Hurley
Maynooth University

https://journal.r-project.org
r-journal@r-project.org

A Computational Analysis of the Dynamics of R Style Based on 108 Million Lines of Code from All CRAN Packages in the Past 21 Years

by Chia-Yi Yen, Mia Huai-Wen Chang, Chung-hong Chan

Abstract The flexibility of R and the diversity of the R community leads to a large number of programming styles applied in R packages. We have analyzed 108 million lines of R code from CRAN and quantified the evolution in popularity of 12 style-elements from 1998 to 2019. We attribute 3 main factors that drive changes in programming style: the effect of style-guides, the effect of introducing new features, and the effect of editors. We observe in the data that a consensus in programming style is forming, such as using lower snake case for function names (e.g. softplus_func) and <- rather than = for assignment.

1 Introduction

R is flexible. For example, one can use <- or = as assignment operators. The following two functions can both be correctly evaluated.

```
sum_of_square <- function(x) {
    return(sum(x^2))
}

sum_oF.square=function(x)
{
    sum(x ^ 2)}
```

One area that can highlight this flexibility is naming conventions. According to the previous research by Bååth (2012), there are at least 6 styles and none of the 6 has dominated the scene. Beyond naming conventions investigated by Bååth (2012), there are style-elements that R programmers have the freedom to adopt, e.g. whether or not to add spaces around infix operators, use double quotation marks or single quotation marks to denote strings. On one hand, these variations provide programmers with freedom. On the other hand, these variations can confuse new programmers and can have dire effects on program comprehension. Also, incompatibility between programming styles might also affect reusability, maintainability (Elish and Offutt, 2002), and open source collaboration (Wang and Hahn, 2017).

Various efforts to standardize the programming style, e.g. Google's R Style Guide (Google, 2019), the Tidyverse Style Guide (Wickham, 2017), Bioconductor Coding Style (Bioconductor, 2015), are available (Table 1) [1].

Among the 3 style-guides, the major differences are the suggested naming convention and indentation, as highlighted in Table 1. Other style-elements are essentially the same. These style guides are based on possible improvement in code quality, e.g. style-elements that improve program comprehension (Oman and Cook, 1991). However, we argue that one should first study the current situation, and preferably, the historical development, of programming style variations (PSV) to supplement these standardization efforts. We have undertaken such a task, so that the larger R communities can have a baseline to evaluate the effectiveness of those standardization efforts. Also, we can have a better understanding of the factors driving increase and decrease in PSV historically, such that more effective standardization efforts can be formulated.

[1] Bååth (2012) lists also Colin Gillespie's R style guide. Additional style guides that we found include the style guides by Henrik Bengtsson, Jean Fan, Iegor Rudnytskyi, Roman Pahl, Paul E. Johnson, Joshua Halls, Datanovia, and daqana. We focus only on the 3 style guides of Tidyverse, Google and Bioconductor is because these 3 are arguably the most influential. There are groups of developers (e.g. contributors to **tidyverse**, Google employees, and Bioconductor contributors) adhering to these 3 styles.

Table 1: Three major style-guides and their differentiating style elements (in Bold): Google, Tidyverse and Bioconductor

Feature	Google	Tidyverse	Bioconductor
Function name	UpperCamel	snake_case	lowerCamel
Assignment	Discourage =	Discourage =	Discourage =
Line length	"limit your code to 80 characters per line"	"limit your code to 80 characters per line"	$\leqslant 80$
Space after a comma	Yes	Yes	Yes
Space around infix operators	Yes	Yes	Yes
Indentation	2 spaces	2 spaces	4 spaces
Integer	Not specified	Not specified (Integers are not explicitly typed in included code examples)	Not specified
Quotes	Double	Double	Not specified
Boolean values	Use TRUE / FALSE	Use TRUE / FALSE	Not specified
Terminate a line with a semicolon	No	No	Not specified
Curly braces	{ same line, then a newline, } on its own line	{ same line, then a newline, } on its own line	Not specified

2 Analysis

Data Source

On July 1, 2020, we cloned a local mirror of CRAN using the rsync method suggested in the CRAN Mirror HOWTO (CRAN, 2019). [2] Our local mirror contains all contributed packages as tarball files (.tar.gz). By all contributed packages, we mean packages actively listed online on the CRAN website as well as orphaned and archived packages. In this analysis, we include all active, orphaned and archived packages.

In order to facilitate the analysis, we have developed the package `baaugwo` (Chan, 2020) to extract all R source code and metadata from these tarballs. In this study, only the source code from the `/R` directory of each tarball file is included. We have also archived the metadata from the `DESCRIPTION` and `NAMESPACE` files from the tarballs.

In order to cancel out the over-representation effect of multiple submissions in a year by a particular package, we have applied the *"one-submission-per-year"* rule to randomly selected only one submission from a year for each package. Unless otherwise noticed, we present below the analysis of this *"one-submission-per-year"* sample. Similarly, unless otherwise noticed, the unit of the analysis is *exported function*. The study period for this study is from 1998 to 2019.

Quantification of PSV

All exported functions in our sample are parsed into a parse tree using the parser from the **lintr** (Hester and Angly, 2019) package.

These parse trees were then filtered for lines with function definition and then linted them using the linters from the lintr package to detect for various style-elements. Style-elements considered in this study are:

fx_assign Use = as assignment operators

[2] Regarding the specification of the hardware used for this analysis, please refer to the README file in our Github repository: `https://github.com/chainsawriot/rstyle`

```
softplusFunc = function(value, leaky = FALSE) {
    if (leaky) {
        warnings("using leaky RELU!")
        return(ifelse(value > 0L, value, value * 0.01))
    }
    return(log(1L + exp(value)))
}
```

fx_opencurly An open curly is on its own line

```
softplusFunc <- function(value, leaky = FALSE)
{
    if (leaky)
    {
        warnings("using leaky RELU!")
        return(ifelse(value > 0L, value, value * 0.01))
    }
    return(log(1L + exp(value)))
}
```

fx_infix No spaces are added around infix operators.

```
softplusFunc<-function(value, leaky=FALSE) {
    if (leaky) {
        warnings("using leaky RELU!")
        return(ifelse(value>0L, value, value*0.01))
    }
    return(log(1L+exp(value)))
}
```

fx_integer Not explicitly type integers

```
softplusFunc <- function(value, leaky = FALSE) {
    if (leaky) {
        warnings("using leaky RELU!")
        return(ifelse(value > 0, value, value * 0.01))
    }
    return(log(1 + exp(value)))
}
```

fx_singleq Use single quotation marks for strings

```
softplusFunc <- function(value, leaky = FALSE) {
    if (leaky) {
        warnings('using leaky RELU!')
        return(ifelse(value > 0L, value, value * 0.01))
    }
    return(log(1L + exp(value)))
}
```

fx_commas No space is added after commas

```
softplusFunc <- function(value,leaky = FALSE) {
    if (leaky) {
        warnings("using leaky RELU!")
        return(ifelse(value > 0L,value,value * 0.01))
    }
    return(log(1L + exp(value)))
}
```

fx_semi Use semicolons to terminate lines

```
softplusFunc <- function(value, leaky = FALSE) {
    if (leaky) {
        warnings("using leaky RELU!");
        return(ifelse(value > 0L, value, value * 0.01));
    }
    return(log(1L + exp(value)));
}
```

fx_t_f Use T/F instead of TRUE / FALSE

```
softplusFunc <- function(value, leaky = F) {
    if (leaky) {
        warnings("using leaky RELU!")
        return(ifelse(value > 0L, value, value * 0.01))
    }
    return(log(1L + exp(value)))
}
```

fx_closecurly An close curly is not on its own line.

```
softplusFunc <- function(value, leaky = FALSE) {
    if (leaky) {
        warnings("using leaky RELU!")
        return(ifelse(value > 0L, value, value * 0.01)) }
    return(log(1L + exp(value))) }
```

fx_tab Use tab to indent

```
softplusFunc <- function(value, leaky = FALSE) {
    if (leaky) {
        warnings("using leaky RELU!")
        return(ifelse(value > 0L, value, value * 0.01))
    }
    return(log(1L + exp(value)))
}
```

We have studied also the naming conventions of all included functions. Using the similar technique of Bååth (2012), we classified function names into the following 7 categories:

- **alllower** softplusfunc
- **ALLUPPER** SOFTPLUSFUNC
- **UpperCamel** SoftPlusFunc
- **lowerCamel** softPlusFunc
- **lower_snake** soft_plus_func
- **dotted.func** soft.plus.func
- **other** sOfTPluSfunc

The last style-element is line-length. For each R file, we counted the distribution of line-length. In this analysis, the unit of analysis is line.

If not considering line-length, the remaining 10 binary and one multinomial leave 7,168 possible combinations of PSVs that a programmer could employ ($7 \times 2^{10} = 7,168$).

Methodology of community-specific variations analysis

On top of the overall patterns based on the analysis of all functions, the community-specific variations are also studied. In this part of the study, we ask the question: do local patterns of PSV exist in various programming communities? To this end, we constructed a dependency graph of CRAN packages by defining a package as a node and an import/suggest relationship as a directed edge. Communities in this dependency graph were extracted using the Walktrap Community Detection Algorithm (Pons

and Latapy, 2005) provided by the **igraph** package (Csardi and Nepusz, 2006). The step parameter was set at 4 for this analysis. Notably, we analyzed the dependency graph as a snapshot, which is built based on the submission history of every package from 1998 to 2019.

By applying the Walktrap Community Detection on the 2019 data, we have identified 931 communities in this CRAN dependency graph. The purpose of this analysis is to show the PSV in different communities. We selected the largest 20 communities for further analysis. The choice of 20 is deemed enough to show these community-specific variations. These 20 identified communities cover 88% of the total 14,491 packages, which shows that the coverage of our analysis is comprehensive. Readers could explore other choices themselves using our openly shared data.

Methodology of within-package variations analysis

As discussed in Gillespie and Lovelace (2016), maintaining a consistent style in source code can enable efficient reading by multiple readers; it is even thought to be a quality of a successful R programmer. In addition to community-level analysis, we extend our work to the package-level, in which we investigate the consistency of different style elements within a package. In this analysis, we studied 12 style elements, including fx_assign, fx_commas, fx_integer, fx_semi, fx_t_f, fx_closecurly, fx_infix, fx_opencurly, fx_singleq, fx_tab, and fx_name. In other words, 11 binary variables (the first 11) and 1 multinomial variable (fx_name) could be assigned to each function within a package.

We quantified the package-level consistency by computing the entropy for each style element. Given a style element S of an R package R_i, with possible n choices $s_1, \ldots s_n$ (e.g. n = 2 for binary; n = 7 for fx_names), the entropy $H(S)$ is calculated as:

$$H(S) = -\sum_{i=1}^{n} P(s_i) \log P(s_i) \tag{1}$$

$P(s_i)$ is calculated as the proportion of all functions in R_i with the style element s_i. For example, if a package has 4 functions and the S of these 4 functions are 0,0,1,2. The entropy $H(S)$ is $-((0.5 \times \log 0.5) + (0.25 \times \log 0.25) + (0.25 \times \log 0.25)) = 0.45$.

As the value of $H(S)$ is not comparable across different S with a different number of n, we normalize the value of $H(S)$ into $H'(S)$ by dividing $H(S)$ with the theoretical maximum. The maximum values of $H(S)$ for n = 2 and n = 7 are 0.693 and 1.946, respectively.

Finally, we calculate the $\overline{H'(S)}$ of all CRAN packages (i.e. $R_1 \ldots R_n$, where n equals the number of all CRAN packages) by averaging the $H'(S)$.

3 Results

We studied more than 108 million lines of code from 17,692 unique packages. In total, 2,249,326 exported functions were studied. Figure 1 displays the popularity of the 10 binary style-elements from 1998 to 2019. Some style-elements have very clear trends towards a majority-vs-minority pattern, e.g. fx_closecurly, fx_semi, fx_t_f and fx_tab. Some styles-elements are instead trending towards a divergence from a previous majority-vs-minority pattern, e.g. fx_assign, fx_commas, fx_infix, fx_integer, fx_opencurly and fx_singleq. There are two style-elements that deserve special scrutiny. Firstly, the variation in fx_assign is an illustrative example of the effect of introducing a new language feature by the R Development Core Team. The introduction of the language feature (= as assignment operator) in R 1.4 (Chambers, 2001) has coincided with the taking off in popularity of such style-element since 2001. Up to now, around 20% of exported functions use such style.

Secondly, the popularity of fx_opencurly shows how a previously established majority style (around 80% in late 90s) slowly reduced into a minority, but still prominent, style (around 30% in late 10s).

Similarly, the evolution of different naming conventions is shown in Figure 2 [3]. This analysis can best be used to illustrate the effect of style-guides. According to Bååth (2012), dotted.func style is very specific to R programming. This style is the most dominant style in the early days of CRAN. However, multiple style guides advise against the use of dotted.func style and thus a significant declining trend

[3] 'Other' is the 4th most popular naming convention. Some examples of function names classified as 'other' are: *Blanc-Sablon*, *returnMessage.maxim*, *table_articles_byAuth*, *mktTime.market*, *smoothed_EM*, *plot.Sncf2D*, *as.igraph.EPOCG*, *TimeMap.new*, *fT.KB*, *IDA_stable*. These functions were classified as 'other' because of the placement of capital letters. For packages using an all capitals object class name (e.g. EPOCG) and S3 generic method names (e.g. as.igraph), their methods are likely to be classified as 'others'. One could also classify these functions as dotted.func. However, we follow both **lintr** and Bååth (2012) to classify a function as dotted.func only when no capital letter is used in its name.

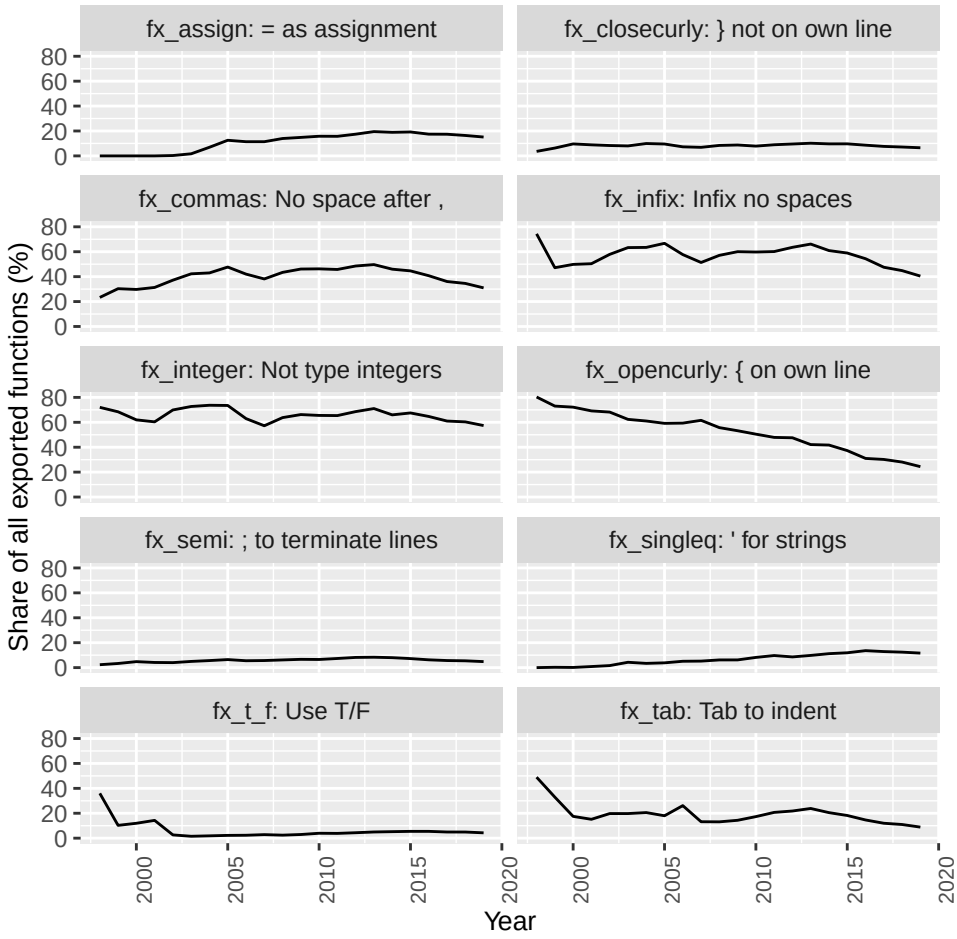


Figure 1: Evolution in popularity of 10 style-elements from 1998 to 2019.

is observed. lower_snake and UpperCamel are the styles endorsed by the Tidyverse Style Guide and the Google's R Style Guide, respectively. These two styles see an increasing trend since the 2010s, while the growth of lower_snake is stronger, with almost a 20% growth in the share of all functions in contrast with the 1-2% growth of other naming conventions. In 2019, lower_snake (a style endorsed by Tidyverse) is the most popular style (26.6%). lowerCamel case, a style endorsed by Bioconductor, is currently the second most popular naming convention (21.3% in 2019). Only 7.0% of functions use UpperCamel, the style endorsed by Google.

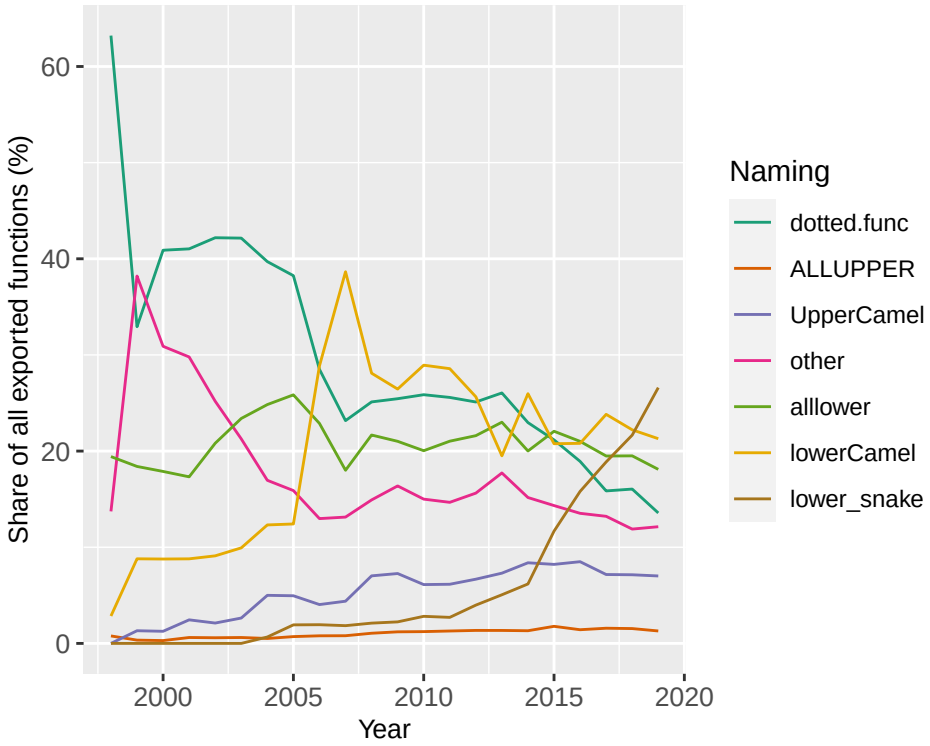


Figure 2: Evolution in popularity of 7 naming conventions from 1998 to 2019.

The evolution of line lengths is tricky to be visualized on a 2-D surface. We have prepared a Shiny app (https://github.com/chainsawriot/rstyle/tree/master/shiny) to visualize the change in line distribution over the span of 21 years. In this paper, Figure 3 shows the snapshot of the change in line length distribution in the range of 40 to 100 characters. In general, developers of newer packages write with less characters per line. Similar to previous analyses with Python programs e.g.Vanderplas (2017), artificial peaks corresponding to recommendations from either style-guides, linters, and editor settings are also observed in our analysis. In 2019, the artificial peak of 80 characters (recommended by most of the style-guides and linters such as **lintr**) is more pronounced for lines with comments but not those with actual code.

Community-based variations

Using the aforementioned community detection algorithm of the dependency graph, the largest 20 communities were extracted. These communities are named by their applications. Table 2 lists the details of these communities [4].

Using the naming convention as an example, there are local patterns in PSV (Figure 4). For example, lower_snake case is the most popular naming convention in the "RStudio" community as expected because it is the naming convention endorsed by the Tidyverse Style-guide. However, only a few functions exported by the packages from "GUI: Gtk" community uses such convention.

For the binary style-elements, local patterns are also observed (Figure 5). The most salient pattern is the exceptional high usage of tab indentation in "rJava" and "Bioinformatics" communities, probably due to influences from Java or Perl. Also, packages in "GUI: Gtk" have an exceptional high usage of open curly on its own line.

[4] "Base packages" (core packages come with R) such as **methods** and **utils** were included in the dependency graph. However, the PSV of recommended packages were not analyzed.

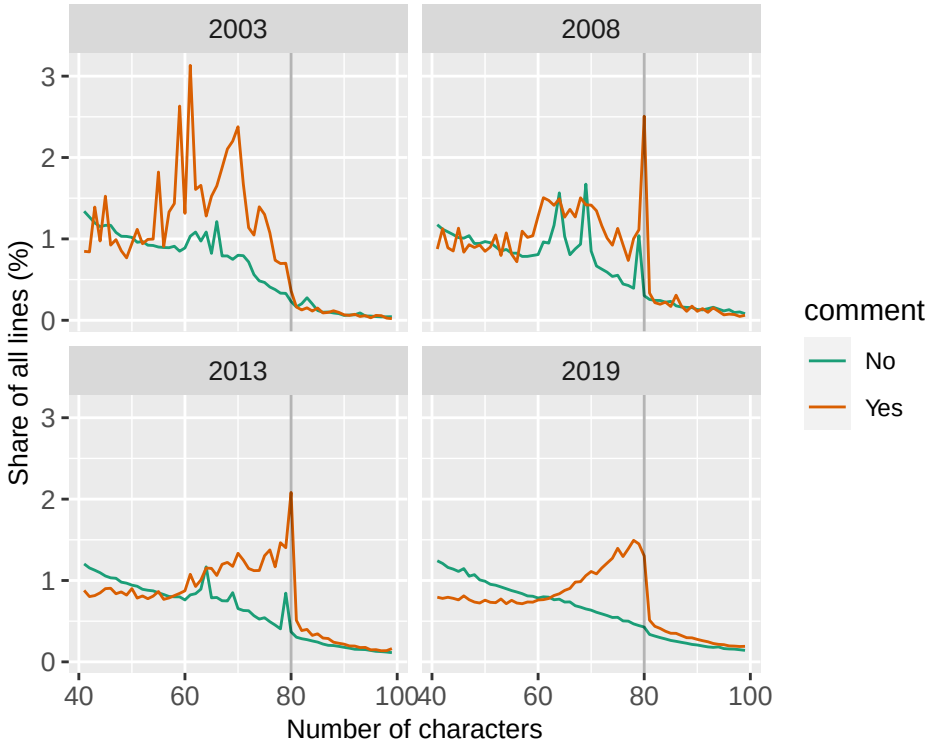


Figure 3: Change in line length distribution of comments (orange) and actual code (green): 2003, 2008, 2013 and 2019.

Table 2: The largest 20 communities and their top 3 packages according to PageRank

Community	Number of Packages	Top 3 Packages
base	5157	methods, stats, MASS
RStudio	4758	testthat, knitr, rmarkdown
Rcpp	826	Rcpp, tinytest, pinp
Statistical Analysis	463	survival, Formula, sandwich
Machine Learning	447	nnet, rpart, randomForest
Geospatial	367	sp, rgdal, maptools
GNU gsl	131	gsl, expint, mnormt
Graph	103	graph, Rgraphviz, bnlearn
Text Analysis	79	tm, SnowballC, NLP
GUI: Tcl/Tk	55	tcltk, tkrplot, tcltk2
Infrastructure	54	rsp, listenv, globals
Numerical Optimization	51	polynom, magic, numbers
Genomics	43	Biostrings, IRanges, S4Vectors
RUnit	38	RUnit, ADGofTest, fAsianOptions
Survival Analysis	33	kinship2, CompQuadForm, coxme
Sparse Matrix	32	slam, ROI, registry
GUI: Gtk	31	RGtk2, gWidgetstcltk, gWidgetsRGtk2
Bioinformatics	29	limma, affy, marray
IO	28	RJSONIO, Rook, base64
rJava	27	rJava, xlsxjars, openNLP

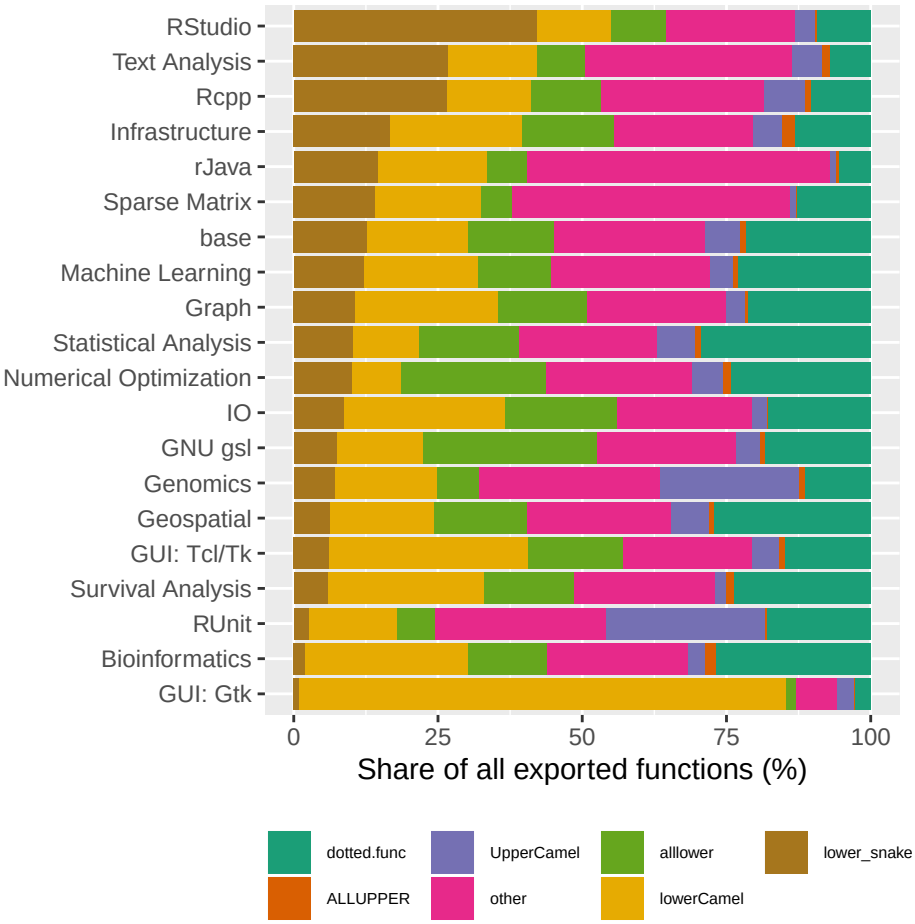


Figure 4: Community-specific distribution of naming conventions among 20 large communities.

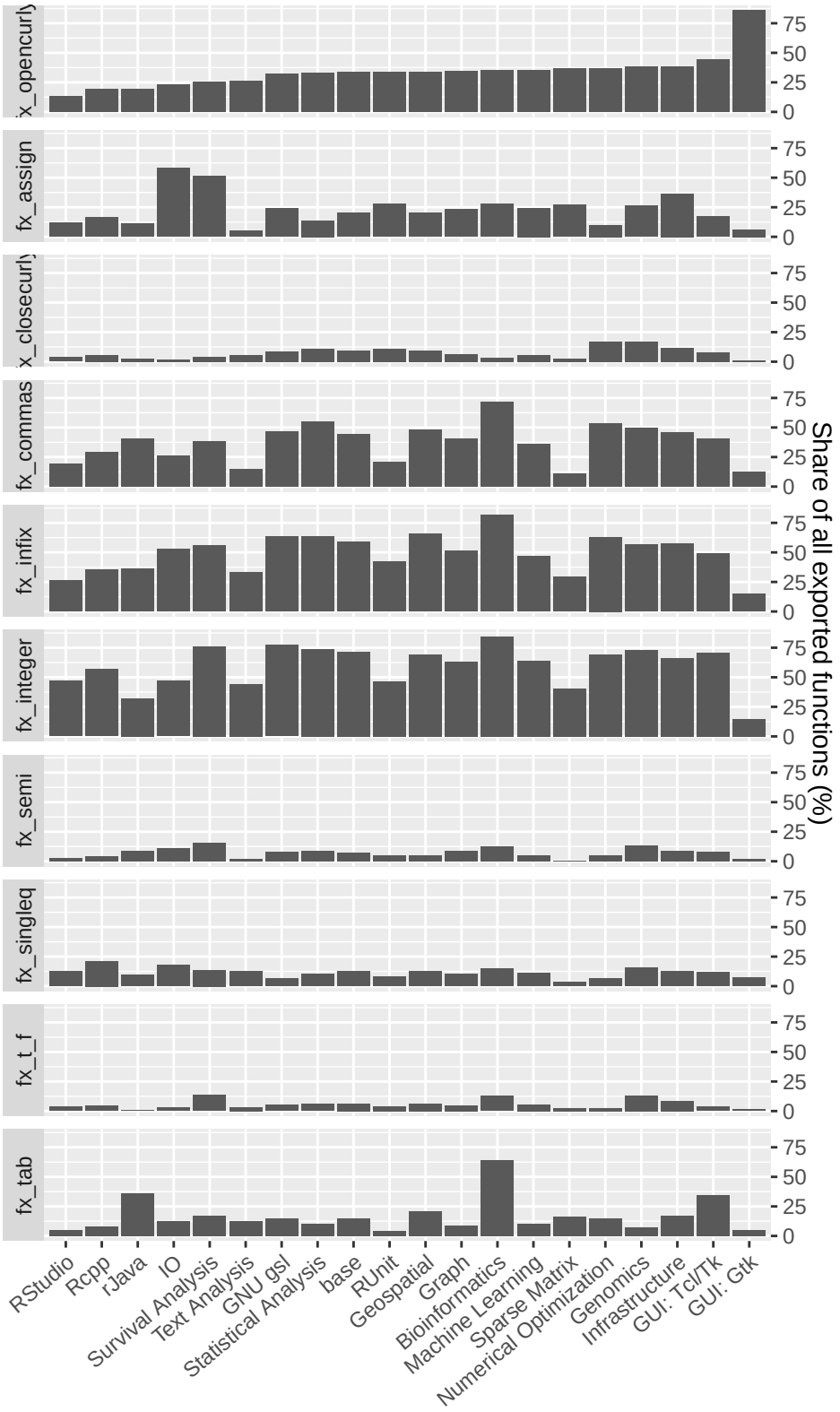


Figure 5: Community-specific distribution of style-elements among 20 large communities

Within-package variations

The result shows that the consistency of style elements within a package varies (Figure 6). For example, style elements like fx_integer, fx_commas, fx_infix, fx_opencurly, and fx_name have less consistency within a package than fx_tab, fx_semi, fx_t_f, fx_closecurly, fx_singleq, and fx_assign. Based on our within-package analysis, we noticed that it is rare for a package to use a consistent style in all of its functions, except those packages with only a few functions. This finding prompts previous concerns e.g. Oman and Cook (1991); Elish and Offutt (2002); Wang and Hahn (2017); Gillespie and Lovelace (2016) that these inconsistent style variations within a software project (e.g. in an R package) might make open source collaboration difficult.

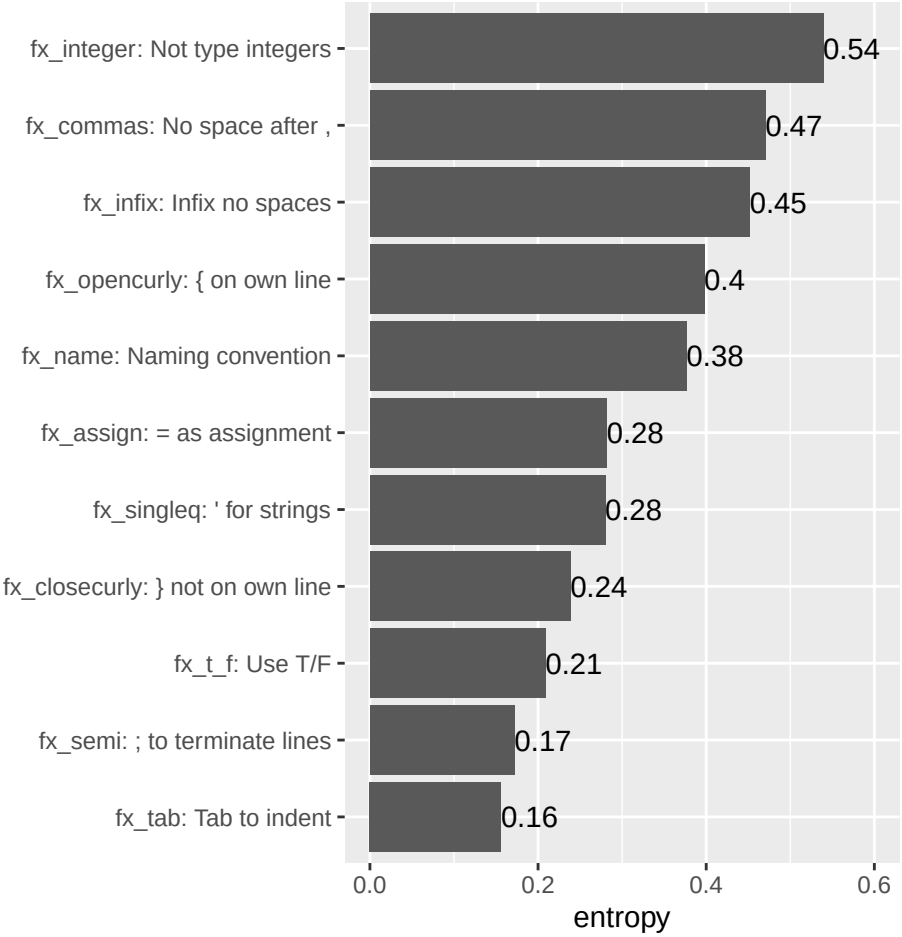


Figure 6: Average package-level entropy of 12 style elements

In Figure 7, we contextualize this finding by showing the distribution of fx_name in 20 R packages with the highest PageRank (Page et al., 1999) in the CRAN dependency graph. Many of these packages have only 1 dominant naming convention (e.g. lower_snake or lowerCamel), but not always. For instance, functions with 6 different naming conventions can be found in the package **Rcpp**.

4 Discussion

In this study, we study the PSV in 21 years of CRAN packages across two dimensions: 1) temporal dimension: the longitudinal changes in popularity of various style-elements over 21 years, and 2) cross-sectional dimension: the variations among communities of the latest snapshot of all packages from 1998 to 2019. From our analysis, we identify three factors that possibly drive PSV: the effect of style-guides (trending of naming conventions endorsed by Wickham (2017) and Google (2019)), the effect of introducing a new language feature (trending of = usage as assignments after 2001), and the effect of editors (the dominance of 80-character line limit).

From a policy recommendation standpoint, our study provides important insight for the R Development Core Team and other stakeholders to improve the current situation of PSV in R. First, the

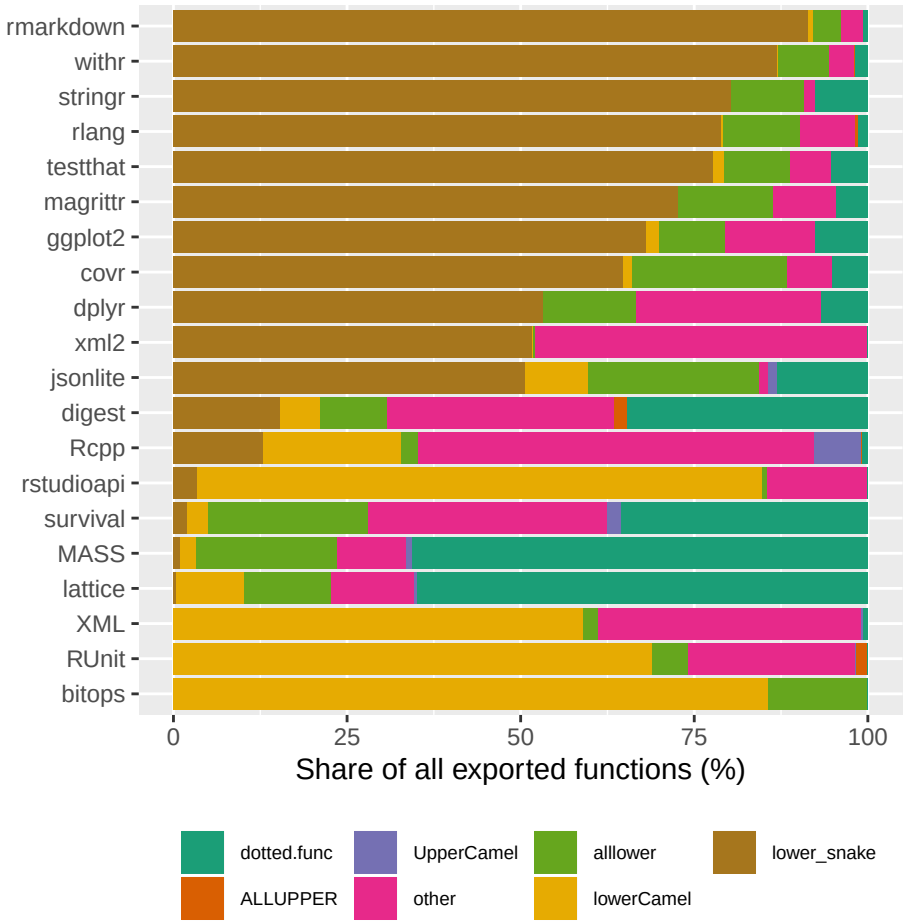


Figure 7: Package-specific distribution of naming conventions among the most important packages

introduction of a new language feature can have a very long-lasting effect on PSV. "Assignments with the = operator" is a feature that introduced by the R Development Core Team to "increase compatibility with S-Plus (as well as with C, Java, and many other languages)" (Chambers, 2001). This might be a good intention but it has an unintended consequence of introducing a very persistent PSV that two major style-guides, Wickham (2017) and Google (2019), consider as a bad style.

Second, style-guides, linters, and editors are important standardizers of PSV. Although we have not directly measured the use of style-guides, linters, and editors in our analysis [5], we infer their effect by studying the time trend (Figure 1). Even with these standardizers, programming styles are slow to change. As indicated by the local PSV patterns, we found in some communities, some package developers have their own style. Having said so, we are not accusing those developers of not following the trendy programming styles. Instead, they follow the mantra of "if it ain't broke don't fix it". Again, from a policy recommendation standpoint, the existence of local PSV patterns suggests there are many blind spots to the previous efforts in addressing PSV. The authors of the style guides may consider community outreach to promote their endorsed styles, if they want other communities to adopt their styles.

Our analysis also opens up an open question: should R adopt an official style-guide akin the PEP-8 of the Python Software Foundation (Van Rossum et al., 2001)? There are of course pros and cons of adopting an official style-guide. As written by Christiansen (1998), "style can easily become a religious issue." It is not our intention to meddle in this "religious issue." If such an effort would be undertaken by someone else, the following consensus-based style could be used as the basis. The following is an example of a function written in such style.

```
softplus_func <- function(value, leaky = FALSE) {
    if (leaky) {
        warnings("using leaky RELU!")
        return(ifelse(value > 0, value, value * 0.01))
    }
    return(log(1 + exp(value)))
}
```

In essence,

- Use snake case
- Use <- to assign, don't use =
- Add a space after commas
- Use TRUE / FALSE, don't use T / F
- Put open curly bracket on same line then a newline
- Use double quotation mark for strings
- Add spaces around infix operators
- Don't terminate lines with semicolon
- Don't explicitly type integers (i.e. 1L)
- Put close curly bracket on its own line
- Don't use tab to indent

We must stress here that this *consensus-based* style is only the most popular style based on our analysis, i.e. the *Zeitgeist* (the spirit of the age) [6]. We have no guarantee that this style can improve clarity or comprehensibility. As a final remark: although enforcing a consistent style can improve open source collaboration (Wang and Hahn, 2017), one must also bear in mind that these rules might need to be adjusted sometimes to cater for programmers with special needs. For example, using spaces instead of tabs for indentation can make code inaccessible to visually impaired programmers (Mosal, 2019).

[5] The usage of style-guides, linters, and editors cannot be directly measured from the record on CRAN. The maintainers usually do not explicitly state the style-guide they endorsed in their code. Similarly, packages that have been processed with linters do not import or suggest linters such as **lintr**, **styler** or **goodpractice**. It might be possible to infer the use of a specific editor such as RStudio in the development version of a package with signals such as the inclusion of an RStudio Project file. These signals, however, were usually removed in the CRAN submission of the package. Future research should use alternative methods to measure the usage of these 3 things in R packages. In this study, similar to other studies, e.g. Bafatakis et al. (2019), we use style compliance as a proxy to usage of a particular style guide, linter or editor.

[6] In 2019, 5.35% of all functions are in this *Zeitgeist* style. Using electoral system as an analogy, this style is having the plurality (have the highest number of votes) but not the absolute majority (have over 50% of the votes)

5 Reproducibility

The data and scripts to reproduce the analysis in this paper are available at https://github.com/chainsawriot/rstyle. An archived version is available at this DOI: http://doi.org/10.5281/zenodo.4026589.

6 Acknowledgments

We have presented a previous version of this paper as a poster at UseR! 2019 Toulouse. Interested readers can access it with the following link: https://github.com/chainsawriot/rstyle/blob/master/docs/Poster_useR2019_Toulouse.png. The work was done prior to Ms Chang joining Amazon Web Services.

The authors would like to thank Wush Wu, Liang-Bo Wang, Taiwan R User group, R-Ladies Taipei, attendees of UseR! 2019, and the two reviewers for their valuable comments that greatly improved this paper.

Bibliography

R. Bååth. The state of naming conventions in R. *The R journal*, 4(2):74–75, 2012. [p6, 9, 10]

N. Bafatakis, N. Boecker, W. Boon, M. Cabello Salazar, J. Krinke, G. Oznacar, and R. White. Python Coding Style Compliance on Stack Overflow. *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, May 2019. doi: 10.1109/msr.2019.00042. URL http://dx.doi.org/10.1109/MSR.2019.00042. [p18]

Bioconductor. *Coding style*, 2015. URL https://www.bioconductor.org/developers/how-to/coding-style/. [p6]

J. Chambers. *Assignments with the = Operator.*, 2001. URL http://developer.r-project.org/equalAssign.html. [p10, 18]

C.-h. Chan. *chainsawriot/baaugwo*, Sept. 2020. URL https://doi.org/10.5281/zenodo.4016596. [p7]

T. Christiansen. *Perl Style: Everyone Has an Opinion*, 1998. URL https://www.perl.com/doc/FMTEYEWTK/style/slide1.html/. [p18]

CRAN. *CRAN Mirror HOWTO/FAQ*, 2019. URL https://cran.r-project.org/mirror-howto.html. [p7]

G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL http://igraph.org. [p10]

M. O. Elish and J. Offutt. The adherence of open source Java programmers to standard coding practices. 2002. [p6, 16]

C. Gillespie and R. Lovelace. *Efficient R programming: a practical guide to smarter programming*. "O'Reilly Media, Inc.", 2016. [p10, 16]

Google. *Google's R Style Guide*, 2019. URL https://google.github.io/styleguide/Rguide.html. [p6, 16, 18]

J. Hester and F. Angly. *lintr: A 'Linter' for R Code*, 2019. URL https://CRAN.R-project.org/package=lintr. R package version 2.0.0. [p7]

C. Mosal. *Nobody talks about the real reason to use Tabs over Spaces.*, 2019. URL https://www.reddit.com/r/javascript/comments/c8drjo/nobody_talks_about_the_real_reason_to_use_tabs/. [p18]

P. W. Oman and C. R. Cook. A programming style taxonomy. *Journal of Systems and Software*, 15(3): 287–301, 1991. [p6, 16]

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. [p16]

P. Pons and M. Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293. Springer, 2005. [p9]

G. Van Rossum, B. Warsaw, and N. Coghlan. PEP 8: style guide for Python code. *Python. org*, 1565, 2001. [p18]

J. Vanderplas. *Exploring Line Lengths in Python Packages*, 2017. URL https://jakevdp.github.io/blog/2017/11/09/exploring-line-lengths-in-python-packages/. [p12]

Z. Wang and J. Hahn. The effects of programming style on open source collaboration. 2017. URL https://aisel.aisnet.org/icis2017/DigitalPlatforms/Presentations/22/. [p6, 16, 18]

H. Wickham. *The tidyverse style guide*, 2017. URL https://style.tidyverse.org/. [p6, 16, 18]

Chia-Yi Yen
Mannheim Business School, Universität Mannheim
L 5, 6, 68131 Mannheim
Germany
https://orcid.org/0000-0003-1209-7789
yen.chiayi@gmail.com

Mia Huai-Wen Chang
Amazon Web Services
Marcel-Breuer-Straße 12, 80807 München
Germany
miachang@amazon.com

Chung-hong Chan
Mannheimer Zentrum für Europäische Sozialforschung, Universität Mannheim
A5, 6, 68159 Mannheim
Germany
https://orcid.org/0000-0002-6232-7530
chung-hong.chan@mzes.uni-mannheim.de

rmonad: pipelines you can compute on

by Zebulun Arendsee, Jennifer Chang, and Eve Syrkin Wurtele

Abstract The **rmonad** package presents a monadic pipeline toolset for chaining functions into stateful, branching pipelines. As functions in the pipeline are run, their results are merged into a graph of all past operations. The resulting structure allows downstream computation on node documentation, intermediate data, performance stats, and any raised messages, warnings or errors, as well as the final results. **rmonad** is a novel approach to designing reproducible, well-documented, and maintainable workflows in R.

1 Background

Pipeline programming is common practice in the R community, with **magrittr**, **pipeR**, and **wrapr** packages offering infix pipe operators (Bache and Wickham, 2014; Ren, 2016; Mount and Zumel, 2018). The value on the left of the pipe operator is passed as the first argument to the right-hand function. This style of programming simplifies code by removing the need to name intermediate values or write deeply nested function calls. For example, using the **magrittr** pipe operator, `%>%`, the expression `x %>% f %>% g` is equivalent to `g(f(x))`. These pipelines are equivalent to applied function compositions and termed function *composition* pipelines.

A *monadic* (Wadler, 1990) pipeline extends composition pipelines by allowing *context* to be threaded through the pipeline. Each function call in the pipeline produces both a new value (assuming successful evaluation) and a computational context surrounding that new value. This new value and context is then merged with the context of the prior node in the pipeline, allowing past context to be stored. In this way, monadic pipelines can be automatically self-describing by returning both the result and a description of the process that created it.

In this paper, we present **rmonad**, the first explicitly monadic pipeline program developed for the R language. **rmonad** captures the history of a pipeline as a graph of all past operations. Each node in the graph represents either an input or a function. These nodes store the source code, documentation, any raised messages/warnings/errors, benchmarking info, and arbitrary additional metadata. **rmonad** also generalizes the standard linear pipeline to a directed graph with support for branching and looping pipelines.

rmonad is one of many graph-based workflow tools available to R programmers. The **drake** package (Landau, 2017) allows specification of R workflows using Make-family semantics (Stallman et al., 2002). The R packages **tidycwl** (Koc et al., 2020) and **sevenbridges** (Xiao and Yin, 2020) wrap the Common Workflow Language which allows specification of DAG-based workflows that can be easily run on high-performance platforms. Many build systems allow execution of R code snippets, such as Snakemake (Köster and Rahmann, 2012), Nextflow (Di Tommaso et al., 2017) and Cuneiform (Brandt et al., 2017). Like these programs, **rmonad** specifies a graph of dependent operations and can handle large, complex projects. However, **rmonad** offers a lighter solution, with no dependencies outside R. In the simplest case, **rmonad** has no more syntactic complexity than a composition pipeline like **magrittr**.

Since **rmonad** can annotate and summarize intermediate data, it can serve as a provenance tracking tool. Provenance tracking of data generated through a pipeline is critical for research reproducibility (Gentleman and Lang, 2007). For example, the provenance manager VisTrails builds directed acyclic graphs (DAG) of workflows and stores intermediate data objects as external XML files in an external database (Silva et al., 2010). It also provides a visualization of the workflow (or provenance trail) as it is being run. By visualizing the workflow in a DAG-like structure, the user can perform exploratory analysis and retooling on the fly. The R provenance tracking packages **archivist** (Biecek and Kosinski, 2017), **trackr** (Becker et al., 2019), and **adapr** (Gelfond et al., 2018) store manual annotations (metadata) of data objects as hooks to an external binary or JSON database.

In the following sections, we introduce the **rmonad** monadic pipeline operator, show how **rmonad** generalizes linear pipelines to support branching and nesting, describe how **rmonad** evaluation allows pipeline debugging and annotation, tie these ideas together with a case study, and provide an overview of the application of **rmonad** to a large-scale project.

2 The monadic pipe operator

A pipeline consists of a series of expressions that are evaluated using upstream data as input. The context that is passed through an **rmonad** pipeline is stored as an "Rmonad" S4 object. This object

consists of a directed graph of the relationships between nodes in the pipelines, a list containing the information about each node (including the output if it is cached), and a unique identifier for the *head* node—the node whose output will be passed to the next operation in the pipeline. Each expression in the pipeline is evaluated by the special **rmonad** function, `evalwrap`, that takes an R expression and returns an "Rmonad" object. After each new expression in a pipeline is evaluated, the past "Rmonad" object is merged with the new one (see Algorithm 1).

```
function evalwrap(x):
    metadata <- get_meta(x)
    doc <- get_doc(x)
    code <- get_code_string(x)
    runtime <- time({ result <- run(x) })
    isOK <- successful(result)
    if isOK then
        y <- result$value
        mem <- size(result$value)
    end
    else
        y <- NULL
        mem <- 0
    end
    return Rmonad(y, isOK, code, metadata, doc, runtime, mem)
```

Algorithm 1: Pseudocode for the **rmonad** eval function, `evalwrap`. `get_meta` and `get_doc` are functions that parse the input expression to extract the documentation string and metadata list. `get_code_string` gets the R code of the function as a string. These three functions rely on the metaprogramming features of R, which allow functions to operate on the code of their inputs. The `run` function is like the standard `eval` R function except that it captures error/warning/message output and returns these together with the output value as a list. `$` is used to access a value in a list. `successful` returns TRUE if the evaluation raised no error. `size` returns the memory footprint of an R object. `Rmonad` is a constructor for an "Rmonad" object. In summary, `evalwrap` evaluates a function call, captures any raised messages, records information about the function and its output, and returns a new "Rmonad" object.

The **rmonad** function `evalwrap` evaluates an R expression and returns an "Rmonad" object. The *type signature* of `evalwrap` is:

$$\text{evalwrap} :: R \to M\ a \tag{1}$$

The `evalwrap` function takes the R expression, R, and returns $M\ a$, which is the "Rmonad" object M wrapping the value returned from the evaluation of R. On success, the returned value has type a. Thus, whereas a composition pipeline would consist of chained functions of type $a \to b$, $b \to c$, $c \to d$, etc, an **rmonad** pipeline consists of $a \to M\ b$, $b \to M\ c$, $c \to M\ d$.

Each evaluation step in an **rmonad** pipeline creates a contextualized object. However, including the context in the output causes a type conflict. For example, suppose there are functions f and g with types $(a \to M\ b)$ and $(b \to M\ c)$, respectively. Function f produces an output of type $M\ b$, but function g requires an input of type b. This conflict is resolved through the special evaluation performed within the monadic pipe operator.

The monadic pipe operator, or the *bind* operator, has the type signature (Wadler, 1990):

$$bind :: \underbrace{m\ b}_{\text{output of } f} \to \underbrace{(b \to m\ c)}_{\text{the function } g} \to \underbrace{m\ c}_{\text{output of } g} \tag{2}$$

where m is a generic monad. The function `bind` takes an input of type $m\ b$ and the function g of type $(b \to m\ c)$. It returns the output of g which has type $m\ c$. Many functions of the general type $a \to m\ b$ can be chained together using this `bind` function. For example, the call `bind(bind(f(x),g),h)` would chain the contextualized results of f through g and then h. The implementation of the bind function defines how context from $m\ b$ is passed through the monadic chain to $m\ c$.

The simplest possible implementation of the `bind` function passes no state and is identical to applied functional composition (e.g., as done in **magrittr**):

function `stateless_bind`(x, g)**:**
  if `successful`(x)**:**
    y = `extract`(x)
    $z = g(y)$
    return z
  else
    return $fail$

Algorithm 2: The bind function for a *composition* pipeline where no context is passed. `successful` returns TRUE if the previous operation succeeded. `extract` returns the stored value from the monadic wrapper. g operates on the y and returns the wrapped value z.

The monadic pipeline operator of **rmonad**, `%>>%`, has the type signature:

$$\underbrace{M\ a}_{lhs} \rightarrow \underbrace{(a \rightarrow\ b)}_{rhs} \rightarrow \underbrace{M\ b}_{output} \tag{3}$$

`%>>%` is a binary operator where the left hand side (`lhs`) is an "Rmonad" object (M) wrapping a value of type a. The right hand side (`rhs`) is a normal R function that takes an input of type a and, if successful, returns a value of type b. If `lhs` stores a failing state (i.e., a prior node in the pipeline raised an error), then the `rhs` function is not evaluated and the failed state is propagated. Otherwise, the value is extracted from `lhs` and `evalwrap` then evaluates the `rhs` function with the `lhs` value as its first argument yielding a new "Rmonad" object. Finally, this new object is merged with the prior, `lhs` "Rmonad" object. Merging involves joining the node graphs of the old and new "Rmonad" objects, setting the head of the resulting graph to the head of the new graph, and removing the value stored in the prior head (see Algorithm 3). The "head" of a graph is critical for branching pipelines (see the Branching and Nesting section).

function `rmonad_bind`(*lhs*, *rhs*)**:**
  h <- `head`(*lhs*)
  if `failed`(*h*) **then**
    return *lhs*
  end
  else
    r2 <- `evalwrap`(*rhs*, `value`(*h*))
    r3 <- `union`(*lhs*, *r2*)
    if `failed`(*r2*) **then**
      r3 <- `set_value`(*r3*, `value`(*h*))
    end
    return *r3*
  end

Algorithm 3: The `%>>%` `bind` function. `lhs` and `rhs` are the left hand side and right hand side of the binary `%>>%` operator, respectively. `lhs` is an "Rmonad" object, which is a graph of past operations. `head` extracts the current node in the graph that is being acted on (the "Rmonad" object stores the index of the current head). `failed` returns TRUE if the operation stored in its argument raised an error. `value` returns the data stored in a node (or in the head node of an "Rmonad" object). `evalwrap` evaluates an R function and its arguments and returns a singleton "Rmonad" object (see Algorithm 1). `union` merges two "Rmonad" objects, assigning the head of the new object to the head of the second object. Here the second "Rmonad" object is a singleton, so we are adding one node to the function graph and making it the new head node. `set_value` sets the value of the head node in an "Rmonad" object. `rmonad_bind` returns a new "Rmonad" object with a new value on success and the old value on failure.

The difference between `%>>%` and a true monadic `bind` operator is that the `rhs` of a monadic bind operator is a function ($a \rightarrow M\ b$), whereas the `rhs` of `%>>%` is a normal R function. The `%>>%` operator essentially transforms the `rhs` R function into a function that yields the monadic object. This is carried out within the monadic bind function through the special evaluation offered by `evalwrap`.

While the primary **rmonad** operator is the monadic pipe operator, `%>>%`, several additional opera-

tors are provided for operating on "Rmonad" objects using pipeline syntax (listed in Table 1).

Operator	Description
`%>>%`	pass `lhs` as initial argument of `rhs` function
`%v>%`	like `%>>%` but caches the `lhs` value
`%*>%`	pass list of arguments from `lhs` to `rhs`
`%__%`	`rhs` starts a new chain that preserves `lhs` history
`%||%`	use `rhs` value if `lhs` is failing
`%|>%`	call `rhs` on `lhs` if `lhs` failed

Table 1: A partial list of the supported operators. `lhs` and `rhs` refer to the left-hand and right-hand sides of the given binary operator. `%>>%` is the primary monadic chain operator. `%v>%` is a variant of the monadic chain operator that always caches its input even on a successful run. The `%*>%` operator takes a list of "Rmonad" objects on the left and feeds the values of each as arguments into the function on the right, linking the history of each input "Rmonad" object to the final "Rmonad" object. This operator is important in building branching pipelines. The `%__%` operator is like a semicolon in a programming language, separating independent pipelines but passing on context. The `%||%` and `%|>%`, operators are used in error recovery.

The `%>>%` operator by itself can only create linear chains of operations. Mechanisms for lifting this limitation are introduced in the next section.

3 Branching and Nesting

In a linear pipeline, the output of each internal function is piped to just one downstream function. In contrast, **rmonad** allows branching to be formed in one of two main ways: 1) the pipeline's head may be reset to an internal node and the pipeline can continue growing from there or 2) multiple pipelines may be merged.

The first branching method uses the `tag` function to attach a label to the current head node and the `view` function to change the head node to a previously tagged node. An example of a branched pipeline using these function is shown in Figure 1. A node may be associated with one or more tags.

The second branching method allows multiple pipelines to merged into one. The most direct merge method uses the `%*>%` operator to pass the head value from each "Rmonad" object in the left-hand side list as arguments to the right-hand side function. **rmonad** also offers a dedicated `loop` function that takes an "Rmonad" object containing a list of values, passes each into monadic function, and connects the histories and final results of each pipeline into a new "Rmonad" node.

The example below demonstrates a loop where nodes where individual elements are dynamically tagged for later access:

```
m <- loop(
  evalwrap(letters[1:3]),
  function(x){ x %>>% paste0("!") %>% tag(c("letters", x)) }
) %*>% paste0
get_value(m, tag="letters")
#> $`letters/c`
#> [1] "c!"
#>
#> $`letters/b`
#> [1] "b!"
#>
#> $`letters/a`
#> [1] "a!"
get_value(m, tag="letters/b")[[1]]
#> "b!"
```

The elements of the first argument to the `loop` function (the letters 'a', 'b', and 'c') are passed to `loop`'s second argument. The second argument is an anonymous function that adds an exclamation mark to the input and tags the resulting value. The tags are hierarchical, thus `get_value(m,tag="letters")` returns all values with the initial tag 'letters'. Specific values can be accessed like files in a path (e.g., "letters/b").

Since **rmonad** pipelines are branched, there is in general no single output value of the pipeline. Rather, the data contained in the "Rmonad" object is queried using a family of vectorized getter

functions. For example, `get_value` will return a list containing the value stored in each node (or `NULL` if no value is stored); `get_error` returns a list of all error messages, `get_warning` returns a list of all warnings, `get_code` returns a list of all code strings, etc. The code below fails on the 'sqrt' call and the failing node can be found by searching for code blocks that were not successfully executed.

```
m <- "a" %>>% paste("cat") %>>% sqrt
get_code(m)[!get_OK(m)]
#> [[1]]
#> [1] "sqrt"
```

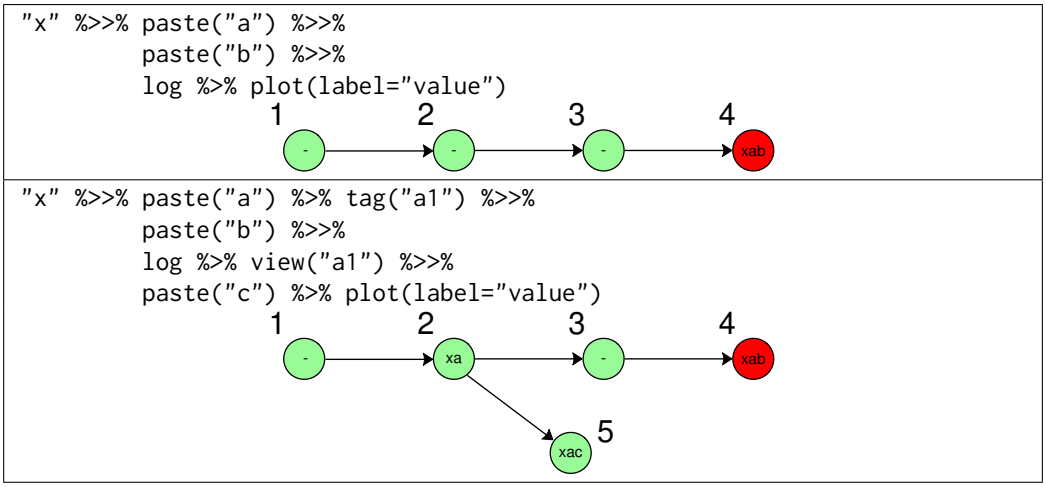


Figure 1: rmonad: linear and branched pipelines. The `plot` functions visualize the graph with values in nodes if the values are cached and "-" otherwise. The layout of the plots was modified in the vector editor Inkscape. **Top:** A linear rmonad pipeline that ends in an error. The pipeline begins at node 1 with the value "x". This is piped into the `paste` function which concatenates the letter "a". Since the `paste` is successful, the result is stored in node 2 and the value in node 1 is deleted to save memory. The value in node 2 is piped into `paste` again, concatenating the letter "b" and storing it in node 3. The value in node 3 is piped into the `log` function, where an error is raised, terminating this branch, and storing the final failing value, "x a b", and the error message. The value is only stored at the end node to avoid storing all intermediate values across a pipeline. That way, values are stored when there are errors or where explicitly tagged by the user. **Bottom:** A branched rmonad pipeline and its resulting graph. From node 2, the "Rmonad" object is piped into the `tag` function which annotates the head node (node 2) with the tag "a1" and sets a flag that ensures the value will be cached for later use. After function 4, the "Rmonad" object is piped into `view`, which sets the head of the graph to node 2. Lastly, the value in node 2 is piped into the final `paste` function that concatenates "c".

In addition to branching, rmonad allows complex pipelines to be built from smaller nested pipelines defined in normal R functions (see Figure 2). When data is piped into a function that wraps a nested rmonad pipeline, the input values will be linked to the nodes in the nested pipeline that use the input. In this way, rmonad enables multilevel debugging. Storing the input to each failed function at each nest level allows a programmer to step through the code in the failed node using the input data, without having to rerun the entire pipeline.

4 Evaluation: error handling, metadata, and post-processing

In this section, we expound on how errors are handled in rmonad, how nodes are documented and annotated, and how post-processing functionality is added to specify log messages, summarize node output and clean up raised messages.

Exception handling and tracebacks

The core functionality of rmonad is the stateful data piping provided by the monadic operator `%>>%`. Linear chains of operations can be constructed with this operator, where each successful node stores information about the function and results. In the case of an error, rmonad provides access to the traceback and to the inputs to each failing function. Knowing the error messages and the function

```
# Level 2
f <- function(x) {
  "<" %v>% paste(x) %v>% paste(">")
}
# Level 1
"A" %v>% paste("B")
    %v>% f
    %v>% paste("C") %>% plot(label="value")
```

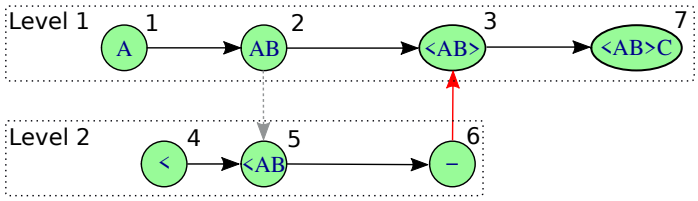


Figure 2: rmonad: complex pipelines can be built from smaller nested pipelines. Level 1 is a pipeline where the node 3 represents the computation performed by the pipeline in Level 2. The nodes contains values ("A", "AB", etc) if the value is cached by rmonad and contains a "-" if the value is not stored. Arrows show relationships between the nodes. A **black** arrow shows data being passed directly to a new function. A **Gray** arrow points from a node in a parent pipeline to a node in a child pipeline that uses its value. The **red** arrow points from the terminal node in a child pipeline to the node in the parent pipeline that stores its result. Stepping through the pipeline: Node 1 wraps the character "A", node 2 appends "B", and node 3 passes "AB" to the function `f`. Next, within the scope of `f`, node 4 starts a new pipeline with the value "<", node 5 pastes "<" from node 4 to the local *x* variable (which is the value passed from node 2), and finally node 6 appends the closing ">" character. The function `f` returns an "Rmonad" object to node 3. The value of node 6 is transferred to node 3 (thus node 6 is empty, "-"). Finally, node 7 appends "C" and the pipeline finishes successfully.

inputs allows the programmer to step through the failed function and easily diagnose the problem. All information is stored within the "Rmonad" object, rather than in the ephemeral state of an R session.

Here is a concrete example:

```
m <- "a cat" %>>% log %>>% sqrt
get_error(m)
#> [[1]]
#> character(0)
#>
#> [[2]]
#> [1] "non-numeric argument to mathematical function"
get_code(m)[[2]]
#> "log"
get_value(m)[[2]]
#> [1] "a cat"
```

Here an illegal value is passed into the natural log function. rmonad catches this error and saves the first failing input and error message. The node index and error message of the failing function can be found with `get_error(m)`, the failing expression can be accessed with `get_code`, and the inputs to the failing function can be retrieved with `get_value`. This approach scales cleanly to large and deeply nested pipelines.

Parsing code strings, docstrings and metadata lists

rmonad leverages R non-standard evaluation to parse the abstract syntax tree of pipeline functions at runtime, prior to evaluation of the functions. rmonad extracts 1) the function's code as a string, 2) an optional documentation string, and 3) an optional list of metadata. All three items are stored in the "Rmonad" node. For example:

```
foo <- function(x){
  "This is a docstring"
  list(sysinfo = sessionInfo())
  return(x)
}
```

The first two lines in the function body are the docstring and metadata list, respectively. Each must 1) be of the appropriate type (string and list, respectively), 2) not be assigned to a variable, and 3) not be the final line in the function body. Thus `foo` is a legal R function that can be used naturally outside of the **rmonad** context. The docstring and metadata would be "dead" lines of code that are evaluated but that are not assigned to any variable or returned. When **rmonad** parses the function before evaluation, the first two lines will be removed and stored, yielding the following function for evaluation:

```
function(x){
  return(x)
}
```

The docstring and the function code are stored as simple strings. The metadata list is evaluated within the function environment, giving it access to function input, and then stored.

The metadata is any list associated with a node. It can be used to store static data such as the author's name, a version for the function, arbitrary notes. It can also store report generation parameters (like code chunks in **knitr**) (Xie, 2015). Because the list is evaluated, its contents are dynamic, allowing, for example, session info to be stored or **knitr** parameters to be a function of the input. Whereas **knitr** nests code chunks and their parameters in a text document, **rmonad** nests text and parameters within the code.

The metadata can be modified freely even *after* the pipeline is run, to enable the user to store notes that are a function of the pipeline results, as well as personal annotations, reminders, or comments on the results.

Post-processing functions: formatting, summarizing, and logging

A built-in use of the metadata is to add formatters, summarizers, and loggers, which are executed automatically after a node is run. For example, a pipeline developer might write the following wrapper around a base 10 log function:

```
fancy_log10 <- function(x){
  list(
    format_warnings = function(x, xs) {
      sprintf("%s NaNs produced", sum(is.na(x)))
    },
    format_log = function(x, passing) {
      if(passing){
        cat("pass\n")
      } else {
        cat("fail\n")
      }
    },
    summarize = list(len = length)
  )
  log10(x)
}
```

When run, the captured warnings are processed by `format_warnings` and log messages by `format_log`, with the following result:

```
"a cat" %>>% fancy_log10 -> m
#> fail
c(-2,-1,0,1,2) %>>% fancy_log10 -> m
#> pass
get_warnings(m)
#> [[1]]
#> character(0)
#>
#> [[2]]
#> [1] "2 NaNs produced"
> get_summary(m)[[2]]$len
#> 5
```

In the first case, an illegal value is passed to the `fancy_log10` function. This leads to a failure in the second node, and the logger prints "fail". In the second case, the user passes the integers between -2 and 2, storing the result in `m`. Since these are legal values (from R's perspective), the logger prints

the message "pass" after evaluation. When the returned object is printed, the post-processed warning message "2 NaNs produced" is shown. The result of the summarizing function is accessed through the `get_summary` function.

5 Case Study: the Iris data

As an example of a simple branching **rmonad** pipeline with error, warning and run time handling we analyzed the Iris dataset (Anderson, 1936; Fisher, 1936). The Iris dataset is often used for case studies of statistics and machine learning workflows, and consists of features of three species of flowers: *Iris setosa*, *Iris virginia*, and *Iris versicolor*. Among these features is petal length. We used three statistical methods, (1) ANOVA, (2) Kruskal-Wallis, and (3) t-test, to determine if petal length is significantly different across the three *Iris* species. Some statistical methods are not appropriate for this dataset without data pre-processing. This case study provides an example of running multiple methods using a branching **rmonad** pipeline, while comparing the output and running times of each method.

Normally, a programmer would run the three methods separately using an R script similar to the following:

```
# === Load data
data(iris)

# === 3 Statistical Tests (run one at a time)
# (1) Anova
res.aov <- aov(Petal.Length ~ Species, data = iris)
summary(res.aov)

# (2) Kruskal-Wallis
res.kr <- kruskal.test(Petal.Length ~ Species, data = iris)
res.kr

# (3) T-Test
t.test(Petal.Length~Species, data=iris)
```

Using **rmonad** tags, data can be branched out to encompass the three statistical tests. Here, the R variable *m* stores the output "Rmonad" S4 object. We must initially tag the branch point node (in this case, the original Iris dataset). Since we gave the first node the tag ("indata"), its value will be cached and can be accessed with the command `get_value(m,tag="indata")`. From here, we can access and pipe (`%>>%`) the viewed "indata" tag into the different statistical tests, as scripted below and visualized in **Figure 3**.

```
# === rmonad (run together)
m <- {
  "iris dataset"
  evalwrap(iris, tag="indata")
} %>>% {
  "anova"
  res.aov <- aov(Petal.Length ~ Species, data = .)
  summary(res.aov)
}

m <- {
  view(m, "indata")
} %>>% {
  "Kruskal-Wallis"
  res.kr <- kruskal.test(Petal.Length ~ Species, data = iris)
  res.kr
}

m <- {
  view(m, "indata")
} %>>% {
  "t-test"
  t.test(Petal.Length~Species, data=iris)
}
```

The above code could have been chained together using `%>% get_value(tag="indata") %>%` commands, but instead was separately added to the *m* rmonad object for ease of reading. From the *m* rmonad objects, we can plot the pipeline. In the following command we label the nodes by node id, documentation, running time, and any errors if they exists.

```
plot(m, label = function(m){paste(get_id(m),
                                  get_doc(m),
                                  get_time(m),
                                  gsub("character\\(0\\)", "", get_error(m)),
                                  sep=":")})
```

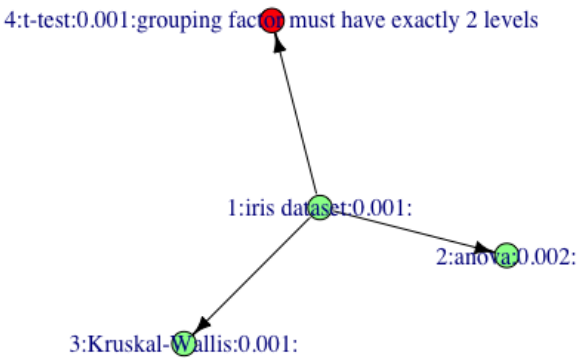


Figure 3: Using rmonad for three statistical tests. The Iris dataset is piped to (1) ANOVA, (2) Kruskul-Wallis, and (3) t-test. Node color reflects whether the test ran (green) or threw an error (red). Time in seconds is shown next to the test name. Errors are annotated on the node. Notice how t-test has the error: "grouping factor must have exactly 2 levels". Of the two tests without errors, ANOVA ran slightly slower than Kruskal-Wallis.

In Figure 3, the center node is the iris dataset and has three arrows going outwards toward one red and two green nodes. Of those, the red node near the top represents the t-test and shows the expected error "grouping factor must have exactly 2 levels". Since we are testing the petal length among the three species, this error is expected. Any errors of the pipeline can also be obtained in a table:

```
missues(m)
#>   id  type                                        issue
#> 1  4 error grouping factor must have exactly 2 levels
```

Going clockwise, ANOVA and Kruskal-Wallis are represented by nodes 2 and 3. The green nodes indicate that both ran although their running times were different. From their node labels, Kruskal-Wallis ran in 0.001 ms, slightly faster than ANOVA (0.002). Also note that green nodes only indicate that the method ran successfully, not the results of that method or statistical significance. The results of the ANOVA and Kruskal-Wallis test can be pulled out of the pipeline using their Node ID number and the following commands.

```
> id=c(2,3)           # place id(s) of end result(s) here
> get_value(m)[id]
#> [[1]]
#>              Df Sum Sq Mean Sq F value Pr(>F)
#> Species       2  437.1  218.55    1180 <2e-16 ***
#> Residuals   147   27.2    0.19
#> ---
#> Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#>
#> [[2]]
#>         Kruskal-Wallis rank sum test
#>
#> data:  Petal.Length by Species
#> Kruskal-Wallis chi-squared = 130.41, df = 2, p-value < 2.2e-16
```

Both tests agree that there is a significant difference between `Petal.Length` across the three Iris species. ANOVA ran on the dataset, which means that petal length follows a normal distribution within each species. Kruskal-Wallis does not assume a normal distribution. The analyst can decide

which method to use; in this case the conclusion is the same. Figure 3 is an example of a branched **rmonad** pipeline comparing three different statistical methods applied to the iris dataset to test a hypothesis.

6 rmonad in the wild: a comparative genomics case study

An example of a large and complex pipeline that uses **rmonad** is the orphan gene classification R pipeline, `fagin` (Arendsee et al., 2019) (Figure 4). This pipeline compares genes from one species of interest (the focal species) to genomes of several related species. The first step in the pipeline is to store the user's session information, which can be used in debugging if needed. Next, the pipeline loops across each species, where, for each species, genomes and annotation data are loaded and validated. Then secondary data (e.g., protein sequences) are derived, and diagnostic summaries are produced and stored. Next, each of the orphan genes in the focal species is compared to each of the related species genomes to create 12 features that are used to classify potential evolutionary relatives of each the focal gene in the target species. Finally, all data for each focal gene is compiled into a description.

The output of this pipeline is a single "Rmonad" object. Further analysis of the pipeline entails a series of queries against this returned object. Warnings and messages are tabulated into an HTML report. Tagged summary data is extracted and used to build diagnostic figures. The primary results are extracted as tabular data and visualized in the final report. Issues with a pipeline can be identified by searching through the raised warnings stored in the "Rmonad" object. Debugging consists of identifying the node of failure, extracting the stored inputs to the failing node, and then stepping through the failing code.

Figure 4: **rmonad** can handle large projects. Here, **rmonad** analysis of the `fagin` pipeline is shown. Green nodes represent passing; orange nodes raise warnings. The four symmetric subtrees on the right represent a loop that loads and validates the input data for four plant species. The two sets of three symmetric subtrees on the left are loops comparing each of the four species (*A. thaliana*) to the other three.

7 Conclusion

We implemented a monadic pipeline in R via the **rmonad** package. **rmonad** provides an infrastructure for data analysis and report generation. **rmonad** stores pipeline results and metadata that can be easily explored interactively and collated into reports using tools such as the literate programming package **knitr** (Xie, 2015) or the HTML report generator **Nozzle.R1** (Gehlenborg et al., 2013).

rmonad integrates a simple profiler into the workflows by automatically capturing the runtime and memory usage of each node. This feature makes it easier for the pipeline developer to identify bottlenecks in the code or potential culprits of memory overflow. Often, a coder must add benchmarking code to key locations in a pipeline. **rmonad** has *built-in* benchmarking, such that all locations in the pipeline are automatically tested and performance can be checked post-run.

rmonad provides a powerful tool for creating and resolving issue reports. If an **rmonad** pipeline fails, the resulting object will store all failing functions, their raised error/warning messages and also their inputs. This object can be used to find the error messages, load all inputs to the failing function, and proceed to step through the code until the bug is found. If the user prepends a node that stores the local session data (e.g., `sessionInfo() %__% ...`), the debugger gains access to the state of the user's machine (an often-requested item in a bug report). An "Rmonad" object with session info attached contains everything needed to debug the issue. This streamlines issue resolution by improving automation and simplifying submission.

Performance has not been a focus of **rmonad** up to this point. The package currently lacks support for the re-use of cached values when pipelines are re-run. Also each evaluation step has a high overhead cost relative to lighter pipeline tools like **magrittr**. **rmonad** pipelines tend to be memory intensive, since they store many intermediate results and metadata in the "Rmonad" objects. Addressing these performance issues is a major goal for future work.

In summary, **rmonad** integrates the concepts of a pipeline, a build system, a data structure, and an low-level report-generating engine. An **rmonad** project consists of incremental piped operations (like a pipeline program), supports complex branching projects (like a build system), and produces a data structure that can be computed on to generate dynamic reports.

8 Availability

rmonad is published under the GPL-3 license and is available on the Comprehensive R Archive Network (CRAN) and on GitHub at `https://github.com/arendsee/rmonad`. Systematic documentation of the features with simple examples can be found in the vignettes, available through CRAN.

9 Funding

This material is based upon work supported by the National Science Foundation under Grant No. IOS 1546858.

Bibliography

E. Anderson. The species problem in iris. *Annals of the Missouri Botanical Garden*, 23(3):457–509, 1936. [p29]

Z. Arendsee, J. Li, U. Singh, P. Bhandary, A. Seetharam, and E. S. Wurtele. fagin: synteny-based phylostratigraphy and finer classification of young genes. *BMC bioinformatics*, 20(1):1–14, 2019. [p31]

S. M. Bache and H. Wickham. *magrittr: A Forward-Pipe Operator for R*, 2014. URL `https://CRAN.R-project.org/package=magrittr`. R package version 1.5. [p22]

G. Becker, S. E. Moore, and M. Lawrence. trackr: a framework for enhancing discoverability and reproducibility of data visualizations and other artifacts in R. *Journal of Computational and Graphical Statistics*, 28(3):644–658, 2019. [p22]

P. Biecek and M. Kosinski. archivist: An R package for managing, recording and restoring data analysis results. *Journal of Statistical Software*, 82(11):1–28, 2017. doi: 10.18637/jss.v082.i11. [p22]

J. Brandt, W. Reisig, and U. Leser. Computation semantics of the functional scientific workflow language cuneiform. *Journal of Functional Programming*, 27, 2017. [p22]

P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame. Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316–319, 2017. [p22]

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2): 179–188, 1936. [p29]

N. Gehlenborg, M. S. Noble, G. Getz, L. Chin, and P. J. Park. Nozzle: a report generation toolkit for data analysis pipelines. *Bioinformatics*, 29(8):1089–1091, 2013. [p31]

J. Gelfond, M. Goros, B. Hernandez, and A. Bokov. A system for an accountable data analysis process in R. *The R journal*, 10 1:6–21, 2018. [p22]

R. Gentleman and D. T. Lang. Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics*, 16:1 – 23, 2007. [p22]

S. Koc, N. Xiao, and D. Dean. *tidycwl: Tidy Common Workflow Language Tools and Workflows*, 2020. URL https://CRAN.R-project.org/package=tidycwl. R package version 1.0.5. [p22]

J. Köster and S. Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28 (19):2520–2522, 2012. [p22]

W. M. Landau. *drake: Data Frames in R for Make*, 2017. URL https://CRAN.R-project.org/package=drake. R package version 4.4.0. [p22]

J. Mount and N. Zumel. Dot-pipe: an s3 extensible pipe for R. *The R Journal*, 2018. [p22]

K. Ren. *pipeR: Multi-Paradigm Pipeline Implementation*, 2016. URL https://CRAN.R-project.org/package=pipeR. R package version 0.6.1.3. [p22]

C. T. Silva, J. Freire, E. Santos, and E. W. Anderson. Provenance-enabled data exploration and visualization with vistrails. *2010 23RD SIBGRAPI - Conference on Graphics, Patterns and Images Tutorials*, pages 1–9, 2010. [p22]

R. M. Stallman, R. McGrath, and P. Smith. *GNU Make: A Program for Directed Compilation*. Free software foundation, 2002. [p22]

P. Wadler. Comprehending monads. In *Proceedings of the 1990 ACM conference on LISP and functional programming*, pages 61–78, 1990. [p22, 23]

N. Xiao and T. Yin. *sevenbridges: Seven Bridges Platform API Client and Common Workflow Language Tool Builder in R*, 2020. https://www.sevenbridges.com, https://sbg.github.io/sevenbridges-r/, https://github.com/sbg/sevenbridges-r. [p22]

Y. Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL https://yihui.name/knitr/. ISBN 978-1498716963. [p28, 31]

Zebulun Arendsee
Iowa State University
Ames IA, USA
https://orcid.org/0000-0002-5833-798X
zbwrnz@gmail.com

Jennifer Chang
Iowa State University
Ames IA, USA
https://orcid.org/0000-0002-8381-3765
jennifer.chang.bioinform@gmail.com

Eve Wurtele
Iowa State University
Ames IA, USA
https://orcid.org/0000-0003-1552-9495
eve@iastate.edu

A Software Tool For Sparse Estimation Of A General Class Of High-dimensional GLMs

by Hassan Pazira, Luigi Augugliaro and Ernst C. Wit

Abstract Generalized linear models are the workhorse of many inferential problems. Also in the modern era with high-dimensional settings, such models have been proven to be effective exploratory tools. Most attention has been paid to Gaussian, binomial and Poisson settings, which have efficient computational implementations and where either the dispersion parameter is largely irrelevant or absent. However, general GLMs have dispersion parameters ϕ that affect the value of the log-likelihood. This in turn, affects the value of various information criteria such as AIC and BIC, and has a considerable impact on the computation and selection of the optimal model.The R-package **dglars** is one of the standard packages to perform high-dimensional analyses for GLMs. Being based on fundamental likelihood considerations, rather than arbitrary penalization, it naturally extends to the general GLM setting. In this paper, we present an improved predictor-corrector (IPC) algorithm for computing the differential geometric least angle regression (dgLARS) solution curve, proposed in Augugliaro et al. (2013) and Pazira et al. (2018). We describe the implementation of a stable estimator of the dispersion parameter proposed in Pazira et al. (2018) for high-dimensional exponential dispersion models. A simulation study is conducted to test the performance of the proposed methods and algorithms. We illustrate the methods using an example. The described improvements have been implemented in a new version of the R-package **dglars**.

1 Introduction

High-dimensional inference problems are studies where the number of predictors p for some response variable is larger than the sample size n. Modern statistical methods developed to study such high-dimensional data are usually based on combining the objective function with a penalty function (i) to calculate a solution curve embedded in the parameter space and then (ii) to find a point on that curve that represents the best compromise between sparsity and predictive behaviour of the model. The recent statistical literature has a great number of contributions devoted to this problem, such as the ℓ_1-penalty function (Tibshirani, 1996), the SCAD method (Fan and Li, 2001) and the Dantzig selector (Candes and Tao, 2007).

Augugliaro et al. (2013) proposed a new approach based on the differential geometrical representation of the likelihood, in particular for a generalized linear model (GLM). The method does not require an explicit penalty function and is called differential geometric LARS (dgLARS) because it generalizes the geometrical ideas on which the least angle regression (Efron et al., 2004) is based. Pazira et al. (2018) extended the dgLARS method to high-dimensional GLMs with exponential dispersion models and arbitrary link functions. In the same paper, the authors repurposed the classic estimation of the dispersion parameter in a high-dimensional setting and also proposed a new, more efficient estimatator. Wit et al. (2020) extended the dgLARS method to sparse inference in relative risk regression models.

From a computational point of view, the main problem of the dgLARS method is related to the standard predictor-corrector (PC) algorithm developed by Augugliaro et al. (2013) to compute the implicitly defined solution path. The PC algorithm becomes computationally intractable when working with thousands of variables because in the prediction step, the number of arithmetic operations needed to compute the Euler predictor scales as the cube of the number of variables. This leads to a cubic increase in the run time needed for computing the solution curve.

In this paper we briefly explain an improved version of the PC algorithm, proposed in Pazira et al. (2018) and Pazira (2020), simply called the improved PC (IPC) algorithm. The IPC algorithm is able to calculate the solution path in fewer, but more relevant points, greatly reducing the computational burden. In addition, we use a much more efficient cyclic coordinate descend (CCD) algorithm (Augugliaro et al., 2012) to calculate a rough dgLARS solution curve for ultra high-dimensional data. In this paper, we focus on the behaviour of the IPC algorithm. The new version of the R-package **dglars** is implemented with both the CCD and IPC algorithms (Augugliaro et al., 2020). The user can also opt to use the old PC algorithm. The package is available on the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=dglars`.

The remaining of this paper is organized as follows. Firstly, we briefly review the differential geometry underlying the dgLARS method and briefly explain the dispersion parameter estimation methods. Next, the new functions implemented in the updated version of the dglars package are

described and shown that they can be used to estimate the dispersion parameter. Then, various simulation studies are performed to evaluate the performance and run times of the proposed estimation algorithms. Finally, we use the functions implemented in the **dglars** package to illustrate its use in an example data set.

2 Methodological Background

In this section we describe very briefly the dgLARS method and the dispersion parameter estimation methods. The interested reader is referred to Augugliaro et al. (2014) and Pazira et al. (2018). In general, the aim of the dgLARS method is to define a continuous model path that attains the highest likelihood with the fewest number of variables.

Geometric foundation and formal definition

Let Y be a scalar random variable with probability density function belonging to the exponential family $p(y;\theta,\phi) = \exp\{(y\theta - b(\theta))/a(\phi) + c(y,\phi)\}$, where $\theta \in \Theta \subseteq \mathcal{R}$ is called canonical parameter, $\phi \in \Phi \subseteq \mathcal{R}^+$ is called dispersion parameter and $a(\cdot)$, $b(\cdot)$ and $c(\cdot,\cdot)$ are specific given functions. We shall assume that Θ is an open set. The expected value of Y is related to the canonical parameter by the mean value mapping, namely $\mathrm{E}(Y) = \mu = \tau(\theta) = \partial b(\theta)/\partial\theta$, where $\tau : int(\Theta) \to \Omega$. Similarly, the variance of Y is related to its expected value by the identity $\mathrm{Var}(Y) = a(\phi)\mathrm{V}(\mu)$, where $\mathrm{V}(\mu)$ is the variance function. Since μ is a reparameterization of the model, in the following paper we denote by $p(y;\mu,\phi)$ the probability density function of Y. Let $\boldsymbol{X}$ be a p-dimensional vector of random predictors, a GLM is based on the assumption that the conditional expected value of Y given $\boldsymbol{X}$ is specified by the identity

$$g(\mathrm{E}(Y|\boldsymbol{X})) = \beta_0 + \sum_{m=1}^{p} x_m \beta_m = \boldsymbol{x}^\top \boldsymbol{\beta},$$

where, with a little abuse of notation, $\boldsymbol{x} = (1, x_1, \ldots, x_p)^\top$ and $g(\cdot)$ is called link function. For notation purposes, it is more convenient to denote $g^{-1}(\boldsymbol{x}^\top\boldsymbol{\beta})$ as $\mu(\boldsymbol{\beta})$.

When we work with n independent and identically distributed copies of the pair $(Y, \boldsymbol{X})$, the marginal distribution of the random vector $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top$ is an element of the set $\mathcal{S} = \{p(\boldsymbol{y};\boldsymbol{\mu},\phi) = \prod_{i=1}^n p(y_i;\mu_i,\phi) : \boldsymbol{\mu} \in \Omega^n, \phi \in \mathcal{R}^+\}$, which is a minimal and regular exponential family of order n and can be treated as a differential manifold in which $\boldsymbol{\mu}$ is a coordinate system (Amari and Nagaoka, 1985). At each point of $\mathcal{S}$ we can attach a tangent space, denoted by $T_{p(\boldsymbol{\mu})}\mathcal{S}$, defined as the as the linear vector space spanned by the n score functions $\partial_i\ell(\boldsymbol{\mu},\phi;\boldsymbol{Y}) = \partial \log p(\boldsymbol{Y};\boldsymbol{\mu},\phi)/\partial\mu_i$. As suggested in Burbea and Rao (1982), each tangent space can be equipped with an inner product: given two tangent vectors belonging to $T_{\boldsymbol{\mu}}\mathcal{S}$, say $v = \sum_{i=1}^n v_i\partial_i\ell(\boldsymbol{\mu},\phi;\boldsymbol{Y})$ and $w = \sum_{i=1}^n w_i\partial_i\ell(\boldsymbol{\mu},\phi;\boldsymbol{Y})$, their inner product is defined to be:

$$\langle v; w\rangle_{\boldsymbol{\mu}} = E_{\boldsymbol{\mu}}(v \cdot w) = \sum_{i=1}^{n} v_i w_i E(\{\partial_i\ell(\boldsymbol{\mu},\phi;\boldsymbol{Y})\}^2) = \sum_{i=1}^{n} \frac{v_i w_i}{a(\phi)V(\mu_i)}. \tag{1}$$

In order to study the geometrical structure of a GLM, we shall assume that $\boldsymbol{\beta} \to \{g^{-1}(\boldsymbol{x}_1^\top\boldsymbol{\beta}), \ldots, g^{-1}(\boldsymbol{x}_n^\top\boldsymbol{\beta})\}^\top = \boldsymbol{\mu}(\boldsymbol{\beta})$ is an embedding, then the set $\mathcal{M} = \{p(\boldsymbol{y};\boldsymbol{\mu}(\boldsymbol{\beta}),\phi) \in \mathcal{S} : \boldsymbol{\beta} \in \mathcal{R}^{p+1}, \phi \in \mathcal{R}^+\}$ is a $p+1$-dimensional submanifold of $\mathcal{S}$. As previously done, the tangent space of $\mathcal{M}$ at the point $p(\boldsymbol{y};\boldsymbol{\mu}(\boldsymbol{\beta}),\phi)$, denoted by $T_{\boldsymbol{\mu}(\boldsymbol{\beta})}\mathcal{M}$, is the linear vector space spanned by the $p+1$ score functions $\partial_h\ell(\boldsymbol{\beta},\phi;\boldsymbol{Y}) = \partial \log p(\boldsymbol{Y};\boldsymbol{\mu}(\boldsymbol{\beta}),\phi)/\partial\beta_h$. Since $T_{\boldsymbol{\mu}(\boldsymbol{\beta})}\mathcal{M}$ is a linear subspace of $T_{\boldsymbol{\mu}(\boldsymbol{\beta})}\mathcal{S}$, the inner product (1) can also be used to define the inner product between two tangent vectors belonging to $T_{\boldsymbol{\mu}(\boldsymbol{\beta})}\mathcal{M}$. For more details see Augugliaro et al. (2013) and Pazira (2017).

The dgLARS estimator is based on a differential geometric characterization of the Rao score test statistic, obtained using the inner product between the bases of the tangent space $T_{\boldsymbol{\mu}(\boldsymbol{\beta})}\mathcal{M}$ and the tangent residual vector $\boldsymbol{r}(\boldsymbol{\beta},\phi,\boldsymbol{y};\boldsymbol{Y}) = \sum_{i=1}^n r_{\boldsymbol{\beta},i}\,\partial_i\ell(\boldsymbol{\beta},\phi;\boldsymbol{Y})$, where $r_{\boldsymbol{\beta},i} = y_i - \mu_i(\boldsymbol{\beta})$. Formally, we have the following identity:

$$\begin{aligned}\partial_h\ell(\boldsymbol{\beta},\phi;\boldsymbol{Y}) &= \langle \partial_h\ell(\boldsymbol{\beta},\phi;\boldsymbol{Y}); \boldsymbol{r}(\boldsymbol{\beta},\phi,\boldsymbol{y};\boldsymbol{Y})\rangle_{\boldsymbol{\mu}(\boldsymbol{\beta})} \\ &= \cos(\rho_h(\boldsymbol{\beta},\phi)) \cdot ||\boldsymbol{r}(\boldsymbol{\beta},\phi,\boldsymbol{y};\boldsymbol{Y})||_{\boldsymbol{\mu}(\boldsymbol{\beta})} \cdot \mathcal{I}_h^{1/2}(\boldsymbol{\beta},\phi),\end{aligned} \tag{2}$$

where $\mathcal{I}_h(\boldsymbol{\beta},\phi)$ is the Fisher information for β_h, and $\rho_h(\boldsymbol{\beta},\phi)$ is a generalization of the Euclidean notion of angle between the h^{th} column of the design matrix and the residual vector $(r_{\boldsymbol{\beta},i})_{i=\{1,2,\ldots,n\}}$. Importantly, (2) shows that the gradient of the log-likelihood function does not generalize the equiangularity condition proposed in Efron et al. (2004) to define the LARS algorithm, since the latter does

not consider the variation related to $\mathcal{I}_h^{1/2}(\boldsymbol{\beta}, \phi)$, which in the case of a GLM is typically not constant. Using the previous identity, one can see that the signed Rao score test statistic, denoted by $r_h(\boldsymbol{\beta}, \phi)$, can be characterized as follows:

$$\begin{aligned} r_h(\boldsymbol{\beta}, \phi) &= \mathcal{I}_h^{-1/2}(\boldsymbol{\beta}, \phi) \cdot \partial_h \ell(\boldsymbol{\beta}, \phi; \boldsymbol{Y}) \\ &= \cos(\rho_h(\boldsymbol{\beta}, \phi)) \cdot \|\boldsymbol{r}(\boldsymbol{\beta}, \phi, \boldsymbol{y}; \boldsymbol{Y})\|_{\boldsymbol{\mu}(\boldsymbol{\beta})}. \end{aligned} \tag{3}$$

From (3) we shall say that two given predictors, say h and k, satisfy the generalized equiangularity condition at the point $(\boldsymbol{\beta}, \phi)$ when $|r_h(\boldsymbol{\beta}, \phi)| = |r_k(\boldsymbol{\beta}, \phi)|$. Inside the dgLARS theory, the generalized equiangularity condition is used to identify the predictors that are included in the active set.

As shown in Pazira et al. (2018), the Rao score test statistic can be written as

$$r_h(\boldsymbol{\beta}, \phi) = (a(\phi))^{-1/2} \frac{\sum_{i=1}^n \partial_h \mu_i(\boldsymbol{\beta}) V^{-1}(\mu_i(\boldsymbol{\beta}))(y_i - \mu_i(\boldsymbol{\beta}))}{\sqrt{\sum_{i=1}^n V^{-1}(\mu_i(\boldsymbol{\beta})) \partial_h \mu_i(\boldsymbol{\beta}))}} = (a(\phi))^{-1/2} r_h(\boldsymbol{\beta}),$$

then the equiangularity condition and the dgLARS method can be defined using only the function $r_h(\boldsymbol{\beta})$.

Formally, the dgLARS is a method for constructing a path of solutions, indexed by a positive parameter γ, where the nonzero estimates of each solution can be defined as follows. For any dataset there exists with probability one a finite decreasing sequence of transitions points, denoted by $\{\gamma^{(j)}\}$. Such that for any $\gamma \in (\gamma^{(j)}; \gamma^{(j-1)})$ the subvector of non-zero estimates, denoted by $\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$, is defined as the solution to the following non-linear equations

$$r_h(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)) - s_h \gamma = 0, \quad \forall\, h \in \mathcal{A}, \tag{4}$$

where $\mathcal{A} = \{h : \hat{\beta}_h(\gamma) \neq 0\}$ is called active set and $s_h = \text{sign}(r_h(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)))$. Furthermore, for any $k \notin \mathcal{A}$ we have that $|r_k(\hat{\boldsymbol{\beta}}(\gamma))| < \gamma$.

At each transition point we have a change in the active set. We shall say that $\gamma^{(j)}$ is an inclusion transition point if there exists $k \notin \mathcal{A}$ such that the equiangularity condition is satisfied, which can also be written as

$$|r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma^{(j)}))| = \gamma^{(j)}. \tag{5}$$

In this case the active set is updated adding the index k, i.e. the predictor X_k is included in the current model. As explained in Augugliaro et al. (2013), a generalization of the lasso estimator can be obtained letting $s_h = \text{sign}(\hat{\beta}_h(\gamma))$, in this way a predictor will be removed from the current model when the sign of the the associated estimate is not in agreement with the sign of the Rao score test statistic. Formally, we shall say that $\gamma^{(j)}$ is an exclusion transition point if there exists $h \in \mathcal{A}$ such that the following condition is satisfied:

$$\text{sign}(r_h(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma^{(j)}))) \neq s_h. \tag{6}$$

In this case the active set is updated removing the index h and X_h is removed from the current model. In Table 1 the pseudo-code of the improved PC algorithm is reported. In order to distinguish between the two generalizations, in this paper the first one is called dgLARS and dgLASSO denotes the generalization of the lasso estimator.

Computational aspects: the improved PC algorithm

Computationally, the problem of how to estimate the dgLARS solution curve can be decomposed into two sub-problems. The first defines an efficient computational method to compute the transition points, i.e., the values of the tuning parameter corresponding to a change in the active set. In other words, at each transition points, say $\gamma^{(j)}$, only one of condition (5) or (6) is satisfied. Note that condition (6) is only used when the generalization of the lasso estimator is considered. The second problem is to define an efficient computational method to compute the path of solutions when $\gamma \in (\gamma^{(j)}; \gamma^{(j-1)})$. This sub-problem requires the solution to the following system of non-linear equations:

$$r_h(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)) - s_h \gamma = 0, \quad \forall\, h \in \mathcal{A},$$

with $\gamma \in (\gamma^{(j)}; \gamma^{(j-1)})$ and where $s_h = \text{sign}(r_h(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)))$ if we want to compute the dgLARS solution curve or $s_h = \text{sign}(\hat{\beta}_h(\gamma))$ if the dgLASSO solution curve is required.

Augugliaro et al. (2013) proposed a predictor-corrector (PC) algorithm to solve the two sub-problems. Although this algorithm can compute the solution curve for moderately large problems, identifying the transition points is extremely inefficient and can led to a significant increase in computational time. This problem is highlighted in Pazira et al. (2018) and Pazira (2020), where an improvement

Table 1: Pseudo-code of the IPC algorithm to compute the dgLARS and dgLASSO solution curves.

Step	Algorithm
1	First compute $\hat{\beta}_0$
2	Set $\mathcal{A} \leftarrow \arg\max_{k \notin \mathcal{A}}\{\lvert r_k(\hat{\beta}_0)\rvert\}$ and $\gamma \leftarrow \lvert r_{h \in \mathcal{A}}(\hat{\beta}_0)\rvert$
3	Repeat
4	Use (8) to compute $\triangle\gamma^{in}$ and set $\triangle\gamma \leftarrow \triangle\gamma^{in}$
5	If method = "dgLASSO" then
6	use (9) and then (10) to compute $\triangle\gamma^{out}$ and $\triangle\bar{\gamma}$, respectively, and
7	set $\triangle\gamma \leftarrow \triangle\bar{\gamma}$
8	Set $\gamma \leftarrow \gamma - \triangle\gamma$
9	Use (7) to compute $\tilde{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$ (*predictor step*)
10	Use $\tilde{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$ as starting point to solve system (4) (*corrector step*)
11	For all $k \notin \mathcal{A}$ compute $r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))$
12	If $\exists k \notin \mathcal{A}$ such that $\left\lvert r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))\right\rvert > \gamma$ then
13	use (11) to compute $\gamma_k^{rf(l)}$ and set $\gamma_k^{rf} \leftarrow \max_l\{\gamma_k^{rf(l)}\}$
14	first set $\triangle\gamma \leftarrow \triangle\bar{\gamma} - (\gamma_k^{rf} - \gamma)$ and then $\gamma \leftarrow \gamma_k^{rf}$, and go to step 9
15	If $\exists k \notin \mathcal{A}$ such that $\left\lvert r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))\right\rvert = \left\lvert r_h(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))\right\rvert$ for all $h \in \mathcal{A}(\gamma)$, then
16	update $\mathcal{A}(\gamma)$
17	Until convergence

to the original PC algorithm is also proposed. In order to make this paper self-contained, we briefly review this algorithm.

Let $\tilde{\boldsymbol{\varphi}}_{\mathcal{A}}(\gamma) = \boldsymbol{\varphi}_{\mathcal{A}}(\gamma) - \boldsymbol{s}_{\mathcal{A}}\gamma$, where $\boldsymbol{\varphi}_{\mathcal{A}}(\gamma) = (r_h(\hat{\boldsymbol{\beta}_{\mathcal{A}}}(\gamma)))_{h\in\mathcal{A}}^{\top}$ and $\boldsymbol{s}_{\mathcal{A}} = (s_h)_{h\in\mathcal{A}}^{\top}$. Suppose we have computed the solution of the system (4) at γ, denoted by $\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$, and we want to compute the next solution at $\gamma - \Delta\gamma \in (\gamma^{(j)}; \gamma^{(j-1)})$. In the predictor step, the new solution is approximated by the following expression:

$$\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma - \Delta\gamma) \approx \tilde{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma - \Delta\gamma) = \hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma) - \Delta\gamma\, \Im_{\mathcal{A}}^{-1}(\gamma)\, \boldsymbol{s}_{\mathcal{A}}, \tag{7}$$

where $\Im_{\mathcal{A}}(\gamma)$ is the Jacobian matrix of the vector function $\boldsymbol{\varphi}_{\mathcal{A}}(\gamma)$ evaluated at $\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$. In the corrector step, the approximation (7) is used as the starting point of the algorithm solving the system of non-linear equations:

$$r_h(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma - \Delta\gamma)) - s_h(\gamma - \Delta\gamma) = 0, \quad \forall\, h \in \mathcal{A}.$$

In order to reduce the computational burden needed to compute the entire path of solutions, $\Delta\gamma$ is chosen in such a way that at $\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma - \Delta\gamma)$ there is a change in the current active set. After straightforward algebra, $\gamma^{(j)}$ can be approximated by $\gamma^{(j-1)} - \Delta\gamma^{\text{in}}$ where the step size $\Delta\gamma^{\text{in}}$ is equal to:

$$\Delta\gamma^{\text{in}} = \min_{k\notin\mathcal{A}} \left\{ \frac{\gamma - r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))}{1 - \mathrm{d}r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))/\mathrm{d}\gamma}; \frac{\gamma + r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))}{1 + \mathrm{d}r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))/\mathrm{d}\gamma} \right\}_{+}. \tag{8}$$

When we want to compute the dgLASSO solution curve, exclusion condition (6) must be added in the computation of the step size. Since the sign of a Rao score test associated with a predictor included in the current model never changes, condition (6) is equivalent to the following condition: $\gamma^{(j)}$ is an exclusion transition point if there exists a $h \in \mathcal{A}$ such that $\hat{\beta}_h(\gamma^{(j)}) = 0$. Combining approximation (7) with the previous condition, it is easy to see that the step size corresponding to the first exclusion can be approximated by the quantity

$$\Delta\gamma^{\text{out}} = \min_{h\in\mathcal{A}}\{\hat{\beta}_h(\gamma)/d_h(\gamma)\}, \tag{9}$$

where $\boldsymbol{d}_{\mathcal{A}}(\gamma) = \Im_{\mathcal{A}}^{-1}(\gamma)\, \boldsymbol{s}_{\mathcal{A}}$. Then the step size for the dgLASSO solution curve can be approximated

by the quantity

$$\Delta\bar{\gamma} = \min\{\Delta\gamma^{\text{in}}; \Delta\gamma^{\text{out}}\}. \tag{10}$$

Since the step size $\Delta\gamma^{\text{in}}$ and $\Delta\gamma^{\text{out}}$ are obtained using the approximation (7), we also include an exclusion step for removing incorrectly included variables in the model. Determining how to implement this exclusion step is the main difference between the PC and IPC algorithms. When an incorrect variable is included in the model after the corrector step, there exists a non-active variable such that the absolute value of the corresponding Rao score test statistic is greater than γ. In this case, the original PC algorithm reduces the step size using a contractor factor cf, whereas the IPC algorithm applies the Regula-Falsi (rf) method. This method uses information about the function $\tilde{\varphi}_k(\gamma) = r_k(\hat{\beta}_{\mathcal{A}}(\gamma)) - \gamma s_k$, draws a secant from $\tilde{\varphi}_k(\gamma_{new})$ to $\tilde{\varphi}_k(\gamma_{old})$, and estimates the root as where it crosses the γ-axis.

From (4), we know that $r_h(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)) - s_h\gamma = 0$ for all $h \in \mathcal{A}$. Indeed, after the corrector step, when there are non-active variables such that the absolute value of the corresponding Rao score test statistic is greater than γ, we want to find a point, γ^{rf} that is close to the true point transition point, reducing the number of the points of the solution curve. It is easy to verify that the root γ_k^{rf} is given by

$$\gamma_k^{rf} = \frac{r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma_{old}))\,\gamma_{new} - r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma_{new}))\,\gamma_{old}}{r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma_{old})) - r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma_{new})) + (\gamma_{new} - \gamma_{old})\,s_k}, \qquad \forall k \notin \mathcal{A} \tag{11}$$

where $s_k = \text{sign}\{r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma_{new}))\}$. Then the optimal step size is defined as

$$\gamma^{rf} = \{\gamma_k^{rf} \;:\; |r_k(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))| > \gamma\}.$$

In total, the main difference between the PC and IPC algorithms is the different techniques used for adjusting the step size to find the transition points. In At the end of next section we examine the performance of the IPC algorithm and compare it to the original PC algorithm by using the functions in the **dglars** package.

Estimation of the dispersion parameter

Since the dispersion parameter ϕ affects the value of the log-likelihood function, it also impacts the value of various information criteria such as AIC and BIC. Therefore, model selection considerations need to take into account the estimation of the dispersion parameter. There are three commonly-used estimators of the dispersion parameter for ordinary GLMs: deviance, maximum likelihood and Pearson estimators (McCullagh and Nelder, 1989). For high-dimensional GLMs, Pazira et al. (2018) proposed two alternative estimators. The first is a generalized version of the Pearson estimator, $\hat{\phi}_P(\gamma)$,

$$\hat{\phi}_P(\gamma) = \frac{1}{n - |\mathcal{A}|} \sum_{i=1}^{n} \frac{(y_i - g^{-1}(\boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)))^2}{V(g^{-1}(\boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)))}. \tag{12}$$

This estimator is fast, but can be improved by the second proposal of an iterative procedure, called General Refitted Cross-Validation (GRCV), to attenuate the influence of irrelevant variables with high spurious correlations.

The idea of the GRCV method is to split the data $(\boldsymbol{y}_n, \boldsymbol{X}_{n\times p})$ randomly into two equal halves $(\boldsymbol{y}_{n_1}^{(1)}, \boldsymbol{X}_{n_1\times p}^{(1)})$ and $(\boldsymbol{y}_{n_2}^{(2)}, \boldsymbol{X}_{n_2\times p}^{(2)})$. Where we assume that the sample size n is even and $n_1 = n_2 = n/2$. In the first stage, the dgLARS method is applied to these two data sets separately to estimate two solution paths $\hat{\boldsymbol{\beta}}_{\mathcal{A}_j}(\gamma)$ based on $(\boldsymbol{y}^{(j)}, \boldsymbol{X}^{(j)})$ where $j = \{1, 2\}$ and $|\mathcal{A}_j| \leq \min(\frac{n}{2} - 1, p)$.

In the second stage, we perform model selection on each training set to determine two small subsets of selected variables $\hat{\mathcal{A}}_1 \subseteq \mathcal{A}_1$ and $\hat{\mathcal{A}}_2 \subseteq \mathcal{A}_2$. To do that, we estimate ϕ by the generalized Pearson estimator (12) on these two data sets separately to obtain valid log-likelihood functions $\ell(\hat{\boldsymbol{\beta}}_{\mathcal{A}_1}(\gamma), \hat{\phi}_P^{(1)}(\gamma); \boldsymbol{y}^{(1)})$ and $\ell(\hat{\boldsymbol{\beta}}_{\mathcal{A}_2}(\gamma), \hat{\phi}_P^{(2)}(\gamma); \boldsymbol{y}^{(2)})$.

In the third stage, the coefficient $\boldsymbol{\beta}$ for each subset of the data are re-estimated using the variables selected on the other subset, i.e., $(\boldsymbol{y}^{(2)}, \boldsymbol{X}_{\hat{\mathcal{A}}_1}^{(2)})$ and $(\boldsymbol{y}^{(1)}, \boldsymbol{X}_{\hat{\mathcal{A}}_2}^{(1)})$. Since the MLE may not always exist, in this stage we propose to use the dgLARS method to estimate the coefficients based on the selected variables $\hat{\boldsymbol{\beta}}_{\hat{\mathcal{A}}_1}(\gamma_0)$ and $\hat{\boldsymbol{\beta}}_{\hat{\mathcal{A}}_2}(\gamma_0)$ where γ_0 is close to zero. If the MLE does exist, then the dgLARS estimate $\hat{\boldsymbol{\beta}}_{\mathcal{A}}(0)$ is equal to the MLE.

Finally, in the fourth stage, we estimate the dispersion parameter ϕ by the following estimator on

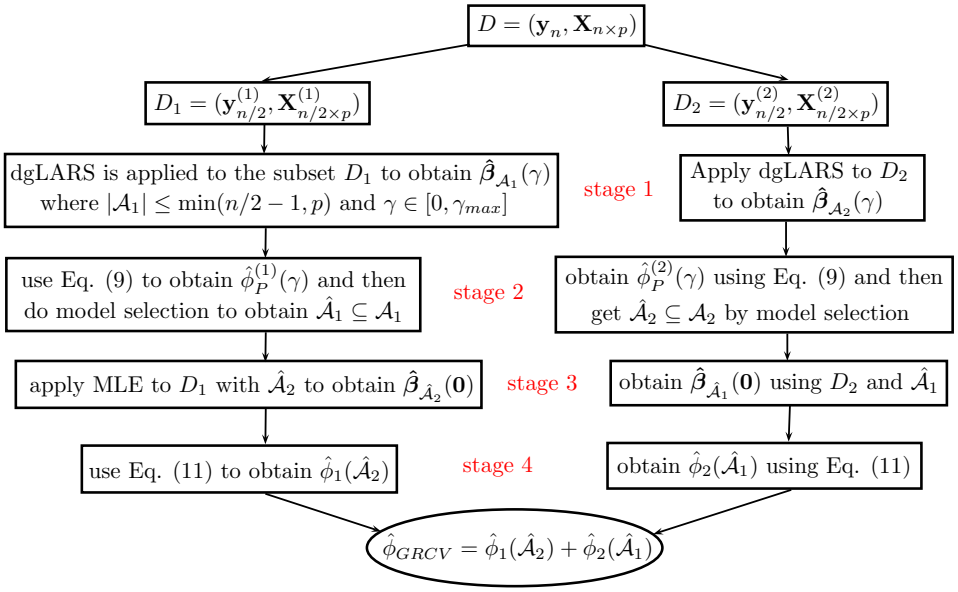


Figure 1: A general diagram for obtaining the GRCV estimate, a four-stage refitted procedure.

the two data sets $(\boldsymbol{y}^{(2)}, \boldsymbol{X}^{(2)}_{\hat{\mathcal{A}}_1})$ and $(\boldsymbol{y}^{(1)}, \boldsymbol{X}^{(1)}_{\hat{\mathcal{A}}_2})$;

$$\hat{\phi}_{GRCV}(\hat{\mathcal{A}}_1, \hat{\mathcal{A}}_2) = \hat{\phi}_1(\hat{\mathcal{A}}_2) + \hat{\phi}_2(\hat{\mathcal{A}}_1), \tag{13}$$

where

$$\hat{\phi}_\zeta(\hat{\mathcal{A}}_j) = \frac{1}{n - 2|\hat{\mathcal{A}}_j|} \sum_{i=1}^{\frac{n}{2}} \frac{\left(y_i^{(\zeta)} - g^{-1}\left((\boldsymbol{x}^{(\zeta)\top}_{i,\hat{\mathcal{A}}_j}\, \hat{\boldsymbol{\beta}}_{\hat{\mathcal{A}}_j}(0)\right)\right)^2}{V\left(g^{-1}\left(\boldsymbol{x}^{(\zeta)\top}_{i,\hat{\mathcal{A}}_j}\, \hat{\boldsymbol{\beta}}_{\hat{\mathcal{A}}_j}(0)\right)\right)}, \quad \zeta \neq j \tag{14}$$

$\boldsymbol{x}^{(\zeta)}_{i,\hat{\mathcal{A}}_j}$ is the i^{th} row of the ζ^{th} subset of the data $\boldsymbol{X}^{(\zeta)}_{\hat{\mathcal{A}}_j}$, $|\hat{\mathcal{A}}_j|$ denotes the cardinality of the set $\hat{\mathcal{A}}_j$, $\hat{\boldsymbol{\beta}}_{\hat{\mathcal{A}}_j}(\gamma)$ is the dgLARS estimator at $\gamma \in [0, \gamma_{max}]$, $\hat{\boldsymbol{\beta}}_{\hat{\mathcal{A}}_j}(0)$ is the ML estimate of $\boldsymbol{\beta}_{\hat{\mathcal{A}}_j}$, $j = \{1,2\}$ and $\zeta = \{1,2\}$.

Figure 1 describes the four step procedure for calculating the GRCV estimate of the dispersion parameter. Since in the second stage of the GRCV procedure the dispersion parameter has to be estimated, an iterative procedure can be defined to reduce its dependence on the generalized Pearson estimator: The algorithm iterates the four steps, such that for the $(\kappa+1)^{\text{th}}$ iteration the κ^{th} GRCV estimate ($\hat{\phi}^{(\kappa)}_{GRCV}$) is used to compute the new $(\kappa+1)^{\text{th}}$ GRCV estimate ($\hat{\phi}^{(\kappa+1)}_{GRCV}$), and so on. Furthermore due to the random cross-validation splits, the estimate contains random variation, and the algorithm will not numerically converge. Therefore, the median of the final iterates can be used as the final GRCV estimate ($\hat{\phi}^{*}_{GRCV}$).

Pazira et al. (2018) showed that the GRCV estimator $\hat{\phi}^{*}_{GRCV}$ is more stable and accurate, which leads to improved overall model selection behaviour.

3 The dglars package: new features

The **dglars** package (Augugliaro et al., 2020) is a collection of computational tools related to the inference procedure of the dgLARS method in the R programming environment.

Description of the new `dglars()` function

Different from the previous version, the new **dglars** package (version 2.1.6) supports the `gaussian`, `binomial`, `poisson`, `Gamma` and `inverse.gaussian` families with the most commonly used link functions. The main function of this package, `dglars()`, is a wrapper function implemented to handle the formula interface usually used in R to create the $n \times p$ model matrix X and the n-dimensional response vector y;

Table 2: Some families and their link functions that can be used in the **dglars** package.

Family	Available link functions
gaussian	"identity", "log", "inverse"
binomial	"logit", "probit", "cauchit", "cloglog", "log"
poisson	"log", "identity", "sqrt"
Gamma	"inverse", "log", "identity"
inverse.gaussian	"1/mu^2", "inverse", "log", "identity"

```
dglars(formula, family = gaussian, g, unpenalized, b_wght, data, subset,
       contrast = NULL, control = list())
```

This function is used to compute the dgLARS/dgLASSO solution curve. As in the standard `glm` function, the user can specify the family and link functions using the argument `family`; see the next section regarding an example of Gamma GLM. This can be a character string naming a family function or the result of a call to a family function. In the new version of the package, the model can be specified by combining family and link functions as described in Table 2. By default the `gaussian` family with `identity` link function is used. In the future, the package will be updated with the `negative.binomial` family with the link functions `log`, `identity`, and `sqrt`.

The argument `control` is a named list of control parameters with the following elements:

```
control = list(algorithm = "pc", method = "dgLASSO", g0 = NULL, nNR= 200,
               nv = NULL, eps = 1.0e-05, np = NULL, dg_max = 0, cf = 0.5,
               NReps = 1.0e-06, ncrct = 50, nccd = 1.0e+05)
```

By using the control parameter `algorithm` it is possible to select the algorithm used to fit the dgLARS solution curve. Setting `algorithm = "pc"` selects the default IPC algorithm; the CCD algorithm is used when `algorithm = "ccd"` is selected. To reduce the computational time needed to compute the dgLARS/dgLASSO solution curve, the algorithms have been written in Fortran 90. The argument `method` is used to choose between the dgLASSO solution curve (`method = "dgLASSO"`) and the dgLARS solution curve (`method = "dgLARS"`).

The `g0` control parameter is used to define the smallest value of the tuning parameter. By default this parameter is set to `1.0e-06` when $p > n$ and to 0.05 otherwise. For more details about the other control parameters and arguments see Augugliaro et al. (2014) and Augugliaro et al. (2020).

When Gaussian, Gamma or inverse Gaussian family is used, `dglars()` returns the vector of estimates for the dispersion parameter; by default, the generalized Pearson statistic is used as estimator but the user can use the function `phihat()` to specify other estimators. For the binomial and Poisson family, the dispersion parameter is assumed known and equal to one.

Description of the functions `grcv()` and `phihat()`

Since the Gaussian, Gamma and inverse Gaussian error distributions have an additional dispersion parameter, this package implements the functions `grcv()` and `phihat()` to estimate the dispersion parameter ϕ for high-dimensional GLMs. The first function implements the method explained in the previous section and can be called as follows:

```
grcv(object, type = c("BIC", "AIC"), nit = 10, control = list(), trace = FALSE, ...)
```

where `object` is a fitted `dglars` object, `type` is the measure of goodness-of-fit used in Step 2 of the algorithm reported in Figure 1. With the current version, the user can choose between the Bayesian (default) and the Akaike information criteria. The argument `nit` is used to specify the number of iterations of the GRCV procedure. The resulting estimate is obtained as the median of the `nit` iterations. `control` is a list of control parameters passed to the function `dglars`, whereas `trace` is a logical variable specifying whether or not information is printed as the GRCV algorithm proceeds. Finally, the argument `...` is used to pass the arguments to the method functions `AIC.dglars` and `BIC.dglars`.

As `grcv()` is only used to estimate the dispersion parameter using the GRCV estimator, the function `phihat()` is specifically developed to handle the all the estimators of the dispersion parameter available in the `dglars` package. This function is defined as follows:

```
phihat(object, type = c("pearson", "deviance", "mle", "grcv"), g = NULL, ...)
```

where `object` is a fitted `dglars` object and `type` is string specifying the estimator of the dispersion parameter. The user can select the Pearson estimator (default), the deviance estimator, the MLE

estimator or the GRCV estimator. The optional argument g is a vector specifying the values of the tuning parameter γ. If not specified (default), the estimates of the dispersion parameter are computed for the sequence of models stored in the argument `object`; for an example see next section. Finally, the argument `...` is used to pass the argument to the function `grcv`. The function `phihat()` returns a vector with the estimates of the dispersion parameter; when `type = "grcv"` all elements of this vector are the same, because the GRCV estimator does not depend on the tuning parameter γ whereas the other three estimators do.

The function `phihat()` is called by the method functions `logLik.dglars()`, `AIC.dglars()` and `coef.dglars()`:

```
logLik(object, phi = c("pearson", "deviance", "mle", "grcv"), g = NULL, ...)

AIC(object, phi = c("pearson", "deviance", "mle", "grcv"), k = 2,
    complexity = c("df", "gdf"), g = NULL, ...)

coef(object, type = c("pearson", "deviance", "mle", "grcv"), g = NULL, ...)
```

when the argument `phi` (or `type` in `coef()`) is set to any of the four estimation methods, i.e., "`pearson`", "`deviance`", "`mle`" or "`grcv`". In the **dglars** package, the `summary()` method:

```
summary(object, type = c("AIC", "BIC"), digits = max(3, getOption("digits") - 3), ...)
```

uses the generalized Pearson estimator to define the BIC or AIC values, but the user can use "`...`" to pass to the method `AIC()` the additional arguments needed to compute a more general measure of goodness-of-fit, e.g., "`phi`", "`k`" or "`complexity`".

An example of a Gamma GLM

To gain more insight about the new features of the **dglars**, we simulated a data set from a Gamma regression model with the `log` link function where the sample size is $n = 50$ and the number of variables is $p = 100$. This is a typical high-dimensional setting ($p > n$). We fix it such that only the first two predictors influence the response variable.

First we install and load the **dglars** package in the R session by the codes

```
R> install.packages("dglars")
R> library("dglars")
```

The corresponding R code is given by:

```
R> set.seed(11235)
R> n <- 50
R> p <- 100
R> s <- 2
R> X <- matrix(runif(n = n * p), n, p)
R> bs <- rep(2, s)
R> Xs <- X[, 1:s]
R> eta <- drop(0.5 + Xs %*% bs)
R> mu <- Gamma("log")$linkinv(eta)
R> shape <- 1
R> phi <- 1 / shape
R> y <- rgamma(n, shape = shape, scale = mu * phi)
R> fit <- dglars(y ~ X, family = Gamma("log"),
+                control = list(algorithm = "pc", method = "dgLARS",
+                g0 = 0.5))
```

We use the argument `g0=0.5` in the function `dglars` to avoid convergence problems coming from the high-dimensionality of the data. The `fit` object is a S3 class '`dglars`', for which the method function `summary.dglars()` can be used to obtain more information about the estimated sequence of models. The following R code shows the output printed by the `summary.dglars()` method with BIC criterion and the GRCV estimate for the dispersion parameter.

```
R> set.seed(11235)
R> summary(fit, type = "BIC", phi = "grcv", control = list(g0 = 0.5))

Call:  dglars(formula = y ~ X, family = Gamma("log"), control = list(algorithm = "pc",
    method = "dgLARS", g0 = 0.5))
```

```
  Sequence       g     %Dev  df    BIC   Rank
           2.5003  0.00000   2  381.6  22
     + X1
           1.9828  0.08380   3  378.4  20
           1.9827  0.08381   3  378.4  19
     + X2
           1.5384  0.20214   4  372.3   8
           1.5314  0.20372   4  372.1   7
    + X12
           1.3876  0.26004   5  371.3   2
           1.3861  0.26060   5  371.2   1 <-
    + X74
           1.2834  0.29734   6  372.0   6
           1.2833  0.29738   6  372.0   5
    + X31
           1.1688  0.33733   7  372.5   9
   + X100
           1.1065  0.36541   8  374.0  10
    + X24
           0.9437  0.44169   9  371.5   4
           0.9413  0.44271   9  371.4   3
    + X71
           0.9208  0.45310  10  374.4  11
     + X9
           0.8460  0.49003  11  375.2  13
           0.8436  0.49117  11  375.1  12
    + X16
           0.7450  0.53586  12  375.2  15
           0.7447  0.53597  12  375.2  14
    + X64
           0.7252  0.54783  13  378.1  18
           0.7250  0.54793  13  378.1  17
    + X18
           0.5902  0.62506  14  375.4  16
     + X6
           0.5821  0.62907  15  379.0  21
    + X36
           0.5659  0.63773  16  382.2  23
    + X37
           0.5279  0.65923  17  384.3  25
           0.5278  0.65929  17  384.3  24
    + X93
           0.5000  0.67501  18  386.8  26

Details:
         BIC values computed using k = 3.912 and complexity = 'df'
         dispersion parameter estimated by 'grcv'

============================================================

Summary of the Selected Model

    Formula: y ~ X1 + X2 + X12
     Family: 'Gamma'
       Link: 'log'

Coefficients:
     Estimate
Int.   1.7494
X1     0.9320
X2     0.5119
X12    0.1749
```

```
Dispersion parameter: 1.044 (estimated by 'grcv' method)
---

                g: 1.386
     Null deviance:  88.74
 Residual deviance:  65.62
              BIC: 371.21

 Algorithm 'pc' ( method = 'dgLARS' )
```

From this output, we can see that the dgLARS method first finds the true predictors (`X1` and `X2`) and then includes the other false predictors. The ranking of the estimated models obtained by the number of estimated non-zero coefficients as a measure of goodness of fit (`complexity = "df"`) is also shown. The corresponding best model is identified by an arrow on the right. The formula of the identified best model, the corresponding estimated coefficients and the estimate of the dispersion parameter are shown in the second section of the output. These values are obtained at the optimal value of the tuning parameter γ, which is calculated by the BIC criterion. For example, from the previous output we can see that the values of the BIC criterion, GRCV estimate and optimal tuning parameter are 371.21, 1.044 and 1.386, respectively. This section shows that GRCV estimate of the dispersion parameter is really close to the true value but the selected model contains false predictors, i.e., `X12`.

Since the deviance, the MLE and the generalized Pearson estimators of the dispersion parameter depend on the tuning parameter γ, the values of these estimates can change during the solution path. The GRCV estimator is computationally more involved, but is fixed across γ. The estimates can be extracted using the `phihat()` function. For example, with the following R code we can see the sequence of values of the tuning parameter with the estimated values of the dispersion parameter by means of the generalized Pearson, deviance, MLE and GRCV methods. For the GRCV method we apply the BIC criterion and `nit=10` iterations inside the algorithm.

```
R> set.seed(11235)
R> g <- fit$g
R> phi.grcv <- phihat(fit, type = "grcv", control = list(g0 = 0.5))
R> phi.pear <- phihat(fit, type = "pearson")
R> phi.dev <- phihat(fit, type = "deviance")
R> phi.mle <- phihat(fit, type = "mle")
R> path <- cbind(g, phi.pear, phi.dev, phi.mle, phi.grcv)


R> print(path, digits = 4)
           g phi.pear phi.dev phi.mle phi.grcv
 [1,] 2.5003   2.2017  1.8111  1.4327    1.044
 [2,] 1.9828   1.9604  1.6939  1.3309    1.044
 [3,] 1.9827   1.9603  1.6938  1.3309    1.044
 [4,] 1.5384   1.6245  1.5065  1.1829    1.044
 [5,] 1.5314   1.6197  1.5035  1.1809    1.044
 [6,] 1.3876   1.4518  1.4275  1.1085    1.044
 [7,] 1.3861   1.4499  1.4264  1.1078    1.044
 [8,] 1.2834   1.3472  1.3857  1.0599    1.044
 [9,] 1.2833   1.3470  1.3856  1.0598    1.044
[10,] 1.1688   1.2357  1.3365  1.0071    1.044
[11,] 1.1065   1.1848  1.3096  0.9696    1.044
[12,] 0.9437   1.0242  1.1797  0.8659    1.044
[13,] 0.9413   1.0218  1.1775  0.8645    1.044
[14,] 0.9208   1.0189  1.1837  0.8502    1.044
[15,] 0.8460   0.9425  1.1314  0.7988    1.044
[16,] 0.8436   0.9394  1.1289  0.7972    1.044
[17,] 0.7450   0.8419  1.0561  0.7340    1.044
[18,] 0.7447   0.8416  1.0559  0.7338    1.044
[19,] 0.7252   0.8336  1.0560  0.7169    1.044
[20,] 0.7250   0.8334  1.0557  0.7167    1.044
[21,] 0.5902   0.6622  0.8993  0.6045    1.044
[22,] 0.5821   0.6704  0.9144  0.5986    1.044
[23,] 0.5659   0.6676  0.9185  0.5858    1.044
[24,] 0.5279   0.6339  0.8894  0.5537    1.044
```

```
[25,] 0.5278   0.6338  0.8893  0.5536    1.044
[26,] 0.5000   0.6160  0.8740  0.5300    1.044
```

By the following R code, we can specify the values of the tuning parameter γ to compute the estimates of the dispersion parameter:

```
R> set.seed(11235)
R> new_g <- seq(range(fit$g)[2], range(fit$g)[1], by = -0.5)
R> phi.grcv <- phihat(fit, g = new_g, type = "grcv",
+                     control = list(g0 = 0.5))
R> phi.pear <- phihat(fit, g = new_g, type = "pearson")
R> phi.dev <- phihat(fit, g = new_g, type = "deviance")
R> phi.mle <- phihat(fit, g = new_g, type = "mle")
R> path <- cbind(new_g, phi.pear, phi.dev, phi.mle, phi.grcv)
R> print(path, digits = 4)
      new_g phi.pear phi.dev phi.mle phi.grcv
[1,] 2.5003   2.2017  1.8111  1.4327    1.044
[2,] 2.0003   1.9677  1.6985  1.3340    1.044
[3,] 1.5003   1.6072  1.5117  1.1647    1.044
[4,] 1.0003   1.0817  1.2328  0.9004    1.044
[5,] 0.5003   0.6163  0.8743  0.5302    1.044
```

Finally, we show the output of function `summary.dglars()` with the generalized Pearson estimator for a comparison with the results yielded by the GRCV method.

```
R> summary(fit, type = "BIC", phi = "pearson")

Call:  dglars(formula = y ~ X, family = Gamma("log"), control = list(algorithm = "pc",
    method = "dgLARS", g0 = 0.5))

  Sequence       g     %Dev  df    BIC   Rank
           2.5003  0.00000   2  382.5  26
     + X1
           1.9828  0.08380   3  380.1  25
           1.9827  0.08381   3  380.1  24
     + X2
           1.5384  0.20214   4  374.4  17
           1.5314  0.20372   4  374.3  16
    + X12
           1.3876  0.26004   5  373.1   8
           1.3861  0.26060   5  373.1   7
    + X74
           1.2834  0.29734   6  373.6  10
           1.2833  0.29738   6  373.6   9
    + X31
           1.1688  0.33733   7  373.7  11
   + X100
           1.1065  0.36541   8  375.0  22
    + X24
           0.9437  0.44169   9  371.3   3
           0.9413  0.44271   9  371.2   2
    + X71
           0.9208  0.45310  10  374.1  14
     + X9
           0.8460  0.49003  11  373.9  13
           0.8436  0.49117  11  373.8  12
    + X16
           0.7450  0.53586  12  372.3   6
           0.7447  0.53597  12  372.3   5
    + X64
           0.7252  0.54783  13  374.9  21
           0.7250  0.54793  13  374.9  20
    + X18
           0.5902  0.62506  14  368.1   1 <-
     + X6
```

```
            0.5821  0.62907  15  371.5   4
    + X36
            0.5659  0.63773  16  374.2  15
    + X37
            0.5279  0.65923  17  374.8  19
            0.5278  0.65929  17  374.8  18
    + X93
            0.5000  0.67501  18  376.3  23

Details:
         BIC values computed using k = 3.912 and complexity = 'df'
         dispersion parameter estimated by 'pearson'

============================================================

Summary of the Selected Model

    Formula: y ~ X1 + X2 + X9 + X12 + X16 + X18 + X24 + X31 + X64 + X71 +
    X74 + X100
     Family: 'Gamma'
       Link: 'log'

Coefficients:
     Estimate
Int.   0.6492
X1     1.6660
X2     1.2259
X9    -0.1183
X12    0.5763
X16   -0.0987
X18   -0.1471
X24    0.6490
X31    0.5249
X64   -0.2859
X71   -0.2110
X74    0.0810
X100  -0.6195

Dispersion parameter: 0.6622 (estimated by 'pearson' method)
---

                g: 0.5902
     Null deviance:  88.74
 Residual deviance:  33.27
              BIC: 368.05

 Algorithm 'pc' ( method = 'dgLARS' )
```

These outputs show that by using different dispersion estimators one can obtain different final models. By using the GRCV estimator, the dgLARS method selects a really small model containing the true predictors, that is `y ~ X1 + X2 + X12`, while using the generalized Pearson estimator our final model contains 12 predictors. We note, however, that the final model selected by the dgLARS method is very sensitive to the (slightly random) value of the GRCV estimator. Although the GRCV tends to work better than the generalized Pearson estimator, no strong conclusions should be attached to this particular example.

Comparing PC and IPC algorithms

In this section we illustrate the difference in performance between the original PC and the new IPC algorithms; for an extensive simulation study see next section. As we mentioned before, the new version of the **dglars** package only implements the IPC and CCD algorithms to compute the dgLARS solution curve. Therefore, we use the PC algorithm in version 1.0.5 of the package (which can only be run using R version 2.10) and the IPC algorithm in the latest version (2.1.6) for the comparisons.

We consider the following R code to simulate a Poisson regression model with the canonical link

function (`link = "log"`), sample size equal to $n = 100$ with $p = 5$ predictors. The corresponding R code is given by:

```
R> set.seed(11235)
R> n <- 100
R> p <- 5
R> X <- matrix(abs(rnorm(n * p)), n, p)
R> b <- 1:2
R> eta <- drop(b[1] + (X[, 1] * b[2]))
R> mu <- poisson()$linkinv(eta)
R> y <- rpois(n, mu)
```

Only the first predictor is set to affect the response variable y. By the following code we estimate the dgLASSO solution curve using the IPC algorithm:

```
R> fit_ipc <- dglars(y ~ X, family = poisson,
+                    control = list(algorithm =  "pc"))
```

By running the following commands we remove the last version and then install the version 1.0.5 of the package to be able to estimate the dgLASSO solution curve using the PC algorithm. The function `install.packages()` can do it for us, such that if the package is already installed, this function replaces it with the specified package from source:

```
R> detach(name = "package:dglars", unload = TRUE)
R> remove.packages(pkgs = "dglars")
R> Old_dglars <- "https://cran.r-project.org/src/contrib/Archive/dglars/
+                dglars_1.0.5.tar.gz"
R> install.packages(Old_dglars, repos = NULL, type = "source")
R> library("dglars")
R> fit_pc <- dglars(y ~ ., family = "poisson",
+                   control =  list(algorithm = "pc"))
```

By printing the 'dglars' object `fit_pc` for our simulated data set, we can see that the number of the points composing the dgLASSO solution curve achieved by the PC algorithm is 25;

```
R> fit_pc

Call:  dglars(formula = y ~ X, family = "poisson", control = list(algorithm = "pc"))

  Sequence        g      Dev    %Dev  df
           68.2417  9403.51  0.0000   1
      +X1
           10.1351   623.36  0.9337   2
            3.7587   186.10  0.9802   2
            2.6310   143.85  0.9847   2
            2.5719   141.99  0.9849   2
            2.5718   141.99  0.9849   2
      +X4
            1.9682   124.04  0.9868   3
            1.6730   116.91  0.9876   3
            1.5270   113.79  0.9879   3
            1.4544   112.34  0.9881   3
            1.4182   111.64  0.9881   3
            1.4001   111.30  0.9882   3
            1.3820   110.96  0.9882   3
      +X3
            1.1309   104.95  0.9888   4
            1.0056   102.37  0.9891   4
            0.9430   101.20  0.9892   4
            0.9117   100.63  0.9893   4
            0.8804   100.09  0.9894   4
      +X2
            0.5796    93.69  0.9900   5
            0.4302    91.44  0.9903   5
            0.3557    90.57  0.9904   5
            0.3186    90.19  0.9904   5
```

```
             0.3000    90.02  0.9904   5
             0.2814    89.85  0.9904   5
       +X5
             0.0001    88.01  0.9906   6

Algorithm pc ( method = dgLASSO ) with exit = 0
```

The number of the iterations computing the solution points by the PC algorithm and the values of the tuning parameter can be obtained by the following code:

```
R> fit_pc$np

[1] 25

R> fit_pc$g

[1]  68.2417321 10.1350645  3.7587453  2.6310292  2.5719482  2.5717722
[7]   1.9681589  1.6729772  1.5269781  1.4543691  1.4181613  1.4000815
[13]  1.3820137  1.1308691  1.0055607  0.9429753  0.9117001  0.8804338
[19]  0.5796210  0.4302023  0.3557410  0.3185725  0.3000037  0.2814428
[25]  0.0001000
```

By printing `fit_ipc`, we can see that the IPC algorithm reduces the number of the iterations for obtaining the solution curve at the change points, leading to significant computational savings.

```
R> fit_ipc

Call:  dglars(formula = y ~ X, family = poisson, control = list(algorithm = "pc"))

  Sequence          g      Dev    %Dev  n. non zero
           68.241732  9403.51  0.0000            1
     + X1
           10.135064   623.36  0.9337            2
            3.758745   186.10  0.9802            2
            2.631029   143.85  0.9847            2
            2.571948   141.99  0.9849            2
            2.571772   141.99  0.9849            2
     + X4
            1.382273   110.97  0.9882            3
            1.382018   110.96  0.9882            3
     + X3
            0.880438   100.09  0.9894            4
     + X2
            0.281457    89.85  0.9904            5
            0.281445    89.85  0.9904            5
     + X5
            0.000001    88.01  0.9906            6

Algorithm 'pc' ( method = 'dgLASSO' ) with exit = 0
```

Fewer than half the number of the iterations are needed by the IPC algorithm compared to the PC algorithm, speeding up the algorithm by a factor of 2.

```
R> fit_ipc$np

[1] 12

R> fit_ipc$g

[1]  6.824173e+01 1.013506e+01 3.758745e+00 2.631029e+00 2.571948e+00
[6]  2.571772e+00 1.382273e+00 1.382018e+00 8.804378e-01 2.814572e-01
[11] 2.814454e-01 9.999996e-07
```

From a computational point of view, the main consequence of using the technique used in the IPC algorithm is a decrease in the run times by adjusting the step size and finding the true transition points. The next section investigates the overall performance of the IPC algorithm by a simulation study.

Table 3: Average CPU times (*time*) in seconds to compute the solution curve using the IPC and PC algorithms based on the logistic regression model, and the mean number of points of the solution curve (q). Standard deviations are in parentheses. The means and standard deviations are trimmed at the 5% level. The IPC algorithm is always faster than the PC algorithm, and the trimmed mean number of q yielded by IPC is always lower than those needed in PC.

		$n = 50$				$n = 200$			
		IPC		PC		IPC		PC	
ρ	p	*time*	q	*time*	q	*time*	q	*time*	q
0.0	100	0.018	52.011	0.022	79.956	0.250	103.74	0.406	190.74
		(0.003)	(5.686)	(0.005)	(12.68)	(0.031)	(5.439)	(0.077)	(16.03)
	1000	0.193	67.622	0.278	99.267	4.333	165.94	5.489	233.06
		(0.023)	(6.225)	(0.056)	(15.51)	(0.458)	(9.120)	(0.935)	(20.51)
	3000	0.775	72.511	0.854	111.00	15.068	183.47	19.933	256.84
		(0.080)	(6.469)	(0.165)	(16.32)	(1.351)	(8.601)	(2.422)	(18.17)
	5000	1.134	68.933	1.205	97.100	24.219	182.83	31.844	249.78
		(0.132)	(6.682)	(0.232)	(16.58)	(2.934)	(11.08)	(5.140)	(22.66)
	7000	1.553	74.378	1.962	109.52	37.149	190.58	49.291	262.67
		(0.190)	(6.604)	(0.444)	(19.62)	(3.198)	(8.760)	(6.439)	(19.96)
0.5	100	0.016	49.167	0.022	80.144	0.174	91.178	0.274	162.03
		(0.003)	(5.613)	(0.004)	(12.57)	(0.022)	(5.170)	(0.049)	(14.02)
	1000	0.150	59.311	0.196	81.611	3.129	143.50	4.116	207.72
		(0.021)	(6.272)	(0.048)	(13.94)	(0.467)	(11.39)	(0.870)	(25.10)
	3000	0.642	65.111	0.684	89.100	10.365	154.49	13.933	216.19
		(0.067)	(7.083)	(0.141)	(16.59)	(1.255)	(9.871)	(2.611)	(23.57)
	5000	1.095	69.122	1.212	98.822	18.663	163.66	25.388	225.16
		(0.126)	(6.505)	(0.235)	(15.34)	(2.355)	(10.87)	(4.095)	(20.52)
	7000	1.420	70.844	1.763	102.00	24.742	159.40	33.101	217.87
		(0.180)	(6.283)	(0.321)	(14.657)	(2.827)	(9.858)	(5.181)	(21.07)

4 Simulation Studies

In this section we present a simulation study to investigate the performance of the improved PC algorithm implemented in the **dglars** package. Although the PC and IPC algorithms compute the same active set, they have different number of arithmetic operations for getting there. The main problem of the PC algorithm is related to the number of the number of arithmetic operations needed to compute the solution curve.

Our simulation study is based on a logistic regression model with sample size equal to $n = 50, 200$. The number of predictors p follows a sequence of five values 100, 1000, 3000, 5000 and 7000. The study is based on two different configurations of the covariance structure of the p predictors, that is, the random vector $X = (X_1, X_2, \cdots, X_p)^\top$ is sampled from an $N(\mathbf{0}, \Sigma)$ distribution with elements of Σ satisfying $corr(X_i; X_j) = \rho^{|i-j|}$, where $\rho = 0$ or $\rho = 0.5$. The response vector is simulated using a model with intercept β_0 and regression coefficients $\boldsymbol{\beta}$ chosen as follows:

$$\beta_0 = 1 \quad \text{and} \quad \boldsymbol{\beta} = (1, 2, 3, \underbrace{0, \cdots, 0}_{p-3}).$$

The R code to replicate our study is reported in the attached file. Table 3 reports the average CPU times in seconds and the mean number of points of the solution curve (q) coming from 100 simulation runs, so that all means and their standard deviations are trimmed of the 5% tails. All timings reported were carried out on a personal computer with Intel Core $i5$ 520M dual-core processor. The proposed IPC algorithm is always faster than the PC algorithm, regardless of the correlation between the predictors. This table also displays that, the trimmed mean number of the points of the solution curve yielded by the IPC algorithm is always lower than those needed in the PC algorithm. Interestingly, when the correlation among the predictors is stronger ($\rho = 0.5$) both algorithms are faster than when there is no correlation. Figure 2 shows the trimmed mean number of the points of the solution curve for the two algorithms. The IPC algorithm is more efficient than the PC algorithm.

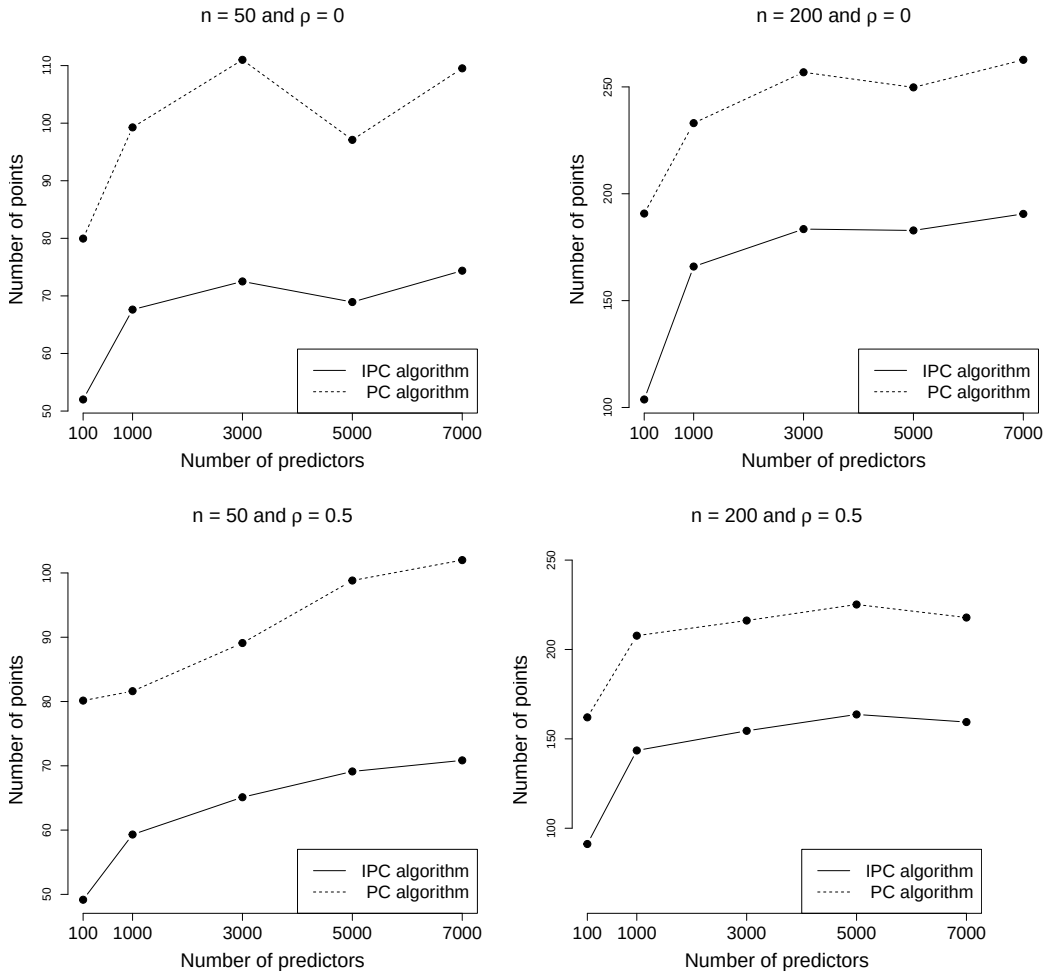


Figure 2: Simulation study for a logistic regression showing the relationship between the trimmed mean number of points of the solution curve q for the IPC and PC algorithms at the 5% level. In all cases, the IPC algorithm is faster than the PC.

5 Application to Example Data

In this section we analyze an example dataset by using the functions available in the **dglars** package. We consider the benchmark `diabetes` data (Efron et al., 2004) to study the sparse structure of an inverse Gaussian regression model. This dataset was also used in Ishwaran et al. (2010) and is available in the R package **lars**:

```
R> install.packages(pkgs = "lars")
R> data("diabetes", package = "lars")
```

The response y are quantitative measurements of disease progression for patients with diabetes after one year. The covariate data include 10 baseline measurements for each patient, recorded in the design matrix `x`, such as *age*, *sex*, *bmi* (body mass index), *map* (mean arterial blood pressure) and six blood serum measurements: *ldl* (low-density lipoprotein), *hdl* (high-density lipoprotein), *ltg* (lamotrigine), *glu* (glucose), *tc* (triglyceride) and *tch* (total cholesterol). In addition to $\binom{10}{2} = 45$ interactions and 9 quadratic terms (excluding the binary sex variable), the design matrix `x2` consists of a total of 64 columns. So, the complete data consists of diabetes progression observations on $n = 442$ patients in combination with $p = 64$ predictor variables. The aim of the study is to identify which of the covariates are important factors in disease progression.

From previous analyses, it was clear that the disease progression is not appropriately modelled by a Gaussian. After some goodness-of-fit considerations, we settle on an inverse Gaussian regression model, which requires us to estimate the dispersion parameter. First, we estimate the optimal value of the tuning parameter γ by 10-fold cross-validation (CV) using the `cvdglars()` function, i.e.,

```
R> library("dglars")
```

```
R> set.seed(11235)
R> cv_diabetes <- cvdglars(y ~ x, family = inverse.gaussian("log"),
+                          data = diabetes)
R> cv_diabetes

Call:  cvdglars(formula = y ~ x, family = inverse.gaussian("log"), data = diabetes)

Coefficients:
     Estimate
Int.   4.9539
xsex  -2.0273
xbmi   2.8447
xmap   2.1969
xtc   -0.3811
xhdl  -2.4124
xltg   3.8501

Dispersion parameter: 0.001141

Details:
   number of non zero estimates: 8
      cross-validation deviance: 0.06224
                              g: 0.01533
                        n. fold: 10

Algorithm 'pc' ( method = 'dgLASSO' )
```

This output shows that the dgLARS method by the help of the CV criterion selects an inverse Gaussian regression model with six covariates (*sex*, *bmi*, *map*, *tc*, *hdl* and *ltg*);

```
R> cv_diabetes$formula_cv
y ~ sex + bmi + map + tc + hdl + ltg
```

Moreover, the optimal tuning parameter is $\gamma = 0.01533$ and the dispersion parameter estimate by the GRCV method is $\hat{\phi}_{GRCV} = 0.001141$. If we had selected the BIC instead of 10-fold cross-validation, we would have obtained

```
R> diabetes_dglars <- dglars(y ~ x, family = inverse.gaussian("log"),
+                            data = diabetes)
R> set.seed(11235)
R> summary(diabetes_dglars, type = "BIC", phi = "grcv")

Call:  dglars(formula = y ~ x, family = inverse.gaussian("log"), data = diabetes)

  Sequence         g     %Dev  df   BIC   Rank
           0.505974  0.00000   2  5223  18
   + xbmi
           0.481473  0.02290   3  5207  17
           0.481262  0.02309   3  5207  16
   + xltg
           0.250152  0.26744   4  4986  15
           0.233248  0.27846   4  4975  14
           0.233174  0.27851   4  4975  13
   + xmap
           0.222313  0.28613   5  4974  12
   + xhdl
           0.100212  0.36560   6  4906  11
           0.099904  0.36572   6  4906  10
   + xsex
           0.030320  0.41322   7  4868   2
           0.030263  0.41324   7  4868   1 <-
    + xtc
           0.014883  0.41892   8  4869   3
   + xglu
           0.005757  0.42063   9  4873   4
```

```
   + xtch
            0.002389  0.42122  10  4879   6
            0.002384  0.42122  10  4879   5
   + xldl
            0.001704  0.42199  11  4884   8
            0.001691  0.42200  11  4884   7
   + xage
            0.000001  0.42272  12  4890   9

Details:
         BIC values computed using k = 6.091 and complexity = 'df'
         dispersion parameter estimated by 'grcv'

============================================================

Summary of the Selected Model

    Formula: y ~ xsex + xbmi + xmap + xhdl + xltg
     Family: 'inverse.gaussian'
       Link: 'log'

Coefficients:
     Estimate
Int.   4.9495
xsex  -1.6834
xbmi   2.7786
xmap   1.9536
xhdl  -2.2917
xltg   3.5420

Dispersion parameter: 0.001112 (estimated by 'grcv' method)
---

                g: 0.03026
     Null deviance:    1.0361
 Residual deviance:    0.6079
               BIC: 4868.0435

 Algorithm 'pc' ( method = 'dgLASSO' )
```

The fitted model now does not include *tc*, but does include the other five predictors (*sex*, *bmi*, *map*, *hdl* and *ltg*). In fact, the optimal value of the tuning parameter $\gamma = 0.03026$ is somewhat larger than with cross-validation, as can be expected from the BIC, resulting in a sparser model. The GRCV estimate of the dispersion parameter $\hat{\phi}_{GRCV} = 001112$, due to the stable nature of the GRCV method. It is also possible to obtain the GRCV estimate directly without a fitted 'dglrs' object, by only using the design matrix x and the response variable y using the following code:

```
set.seed(11235)
grcv(diabetes_dglars, type = "BIC")

[1] 0.001111604
```

Since the original PC algorithm is only available for the version 1.0.5 (and older) and the inverse Gaussian family has only been added to the package from version 2.0.0 onwards, we are not able to compare the run times and also the number of the iterations computing the solution points for the PC and IPC algorithms. The run time of the IPC algorithm is given using the following R code:

```
R> system.time(diabetes_dglars_ipc <- dglars(y ~ x,
+              family = inverse.gaussian("log"), data = diabetes))

user  system elapsed
0.016   0.000   0.016
```

and the number of iterations computing the solution points by the IPC algorithm is

```
R> diabetes_dglars_ipc$np

[1] 18
```

6 Conclusions

In this paper, we have described improvements to the R package **dglars** for estimating a larger class of generalized linear models with arbitrary link functions and a general class of exponential dispersion models. We briefly reviewed the differential geometrical theory underlying the dgLARS method and briefly explained the dispersion parameter estimation methods. We described some functions implemented in the new version of the **dglars** package that can be used to estimate the dispersion parameter. We also used these functions to compare run times between the new IPC and old PC algorithms. The simulations showed that the IPC algorithm is significantly faster than the original PC algorithm. The new version of **dglars** is available on CRAN.

Acknowledgments

This article is based upon work from COST Action *European Cooperation for Statistics of Network Data Science* (COSTNET, CA15109), supported by COST (European Cooperation in Science and Technology).

Bibliography

S. I. Amari and H. Nagaoka. *Differential-geometrical Methods in Statistics*. Springer, New York, 1985. [p35]

L. Augugliaro, A. M. Mineo, and E. C. Wit. Differential geometric lars via cyclic coordinate descent method. *International Conference on Computational Statistics (COMPSTAT 2012)*, pages 67–79, 2012. Limassol, Cyprus. [p34]

L. Augugliaro, A. M. Mineo, and E. C. Wit. Differential geometric least angle regression: A differential geometric approach to sparse generalized linear models. *Journal of the Royal Statistical Society: Series B*, 75(3):471–498, 2013. [p34, 35, 36]

L. Augugliaro, A. M. Mineo, and E. C. Wit. dglars: An R package to estimate sparse generalized linear models. *Journal of Statistical Software*, 59(8):1–40, 2014. [p35, 40]

L. Augugliaro, A. M. Mineo, E. C. Wit, and H. Pazira. *dglars: Differential Geometric LARS Method*. R package version 2.1.6, 2020. http://CRAN.R-project.org/package=dglars. [p34, 39, 40]

J. Burbea and R. C. Rao. Entropy differential metric, distance and divergence measures in probability spaces - a unified approach. *Journal of Multivariate Analysis*, 12:575–596, 1982. [p35]

E. J. Candes and T. Tao. The dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics*, 35:2313–2351, 2007. [p34]

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2): 407–499, 2004. [p34, 35, 49]

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001. [p34]

H. Ishwaran, U. B. Kogalur, and J. Rao. spikeslab: Prediction and variable selection using spike and slab regression. *The R Journal*, 2(2):68–73, 2010. [p49]

P. McCullagh and J. A. Nelder. *Generalized Liner Models*. Chapman & Hall, London, 2^{nd} edition, 1989. [p38]

H. Pazira. *High-dimensional variable selection for GLMs and survival models*. PhD thesis, University of Groningen, 2017. URL https://pure.rug.nl/ws/portalfiles/portal/44052989/Complete_thesis.pdf. [p35]

H. Pazira. Improved predictor-corrector algorithm. In *Computational Intelligence Methods for Bioinformatics and Biostatistics*, pages 99–106. Springer International Publishing, 2020. ISBN 978-3-030-34585-3. URL https://link.springer.com/chapter/10.1007/978-3-030-34585-3_9. [p34, 36]

H. Pazira, L. Augugliaro, and E. Wit. Extended differential geometric lars for high-dimensional glms with general dispersion parameter. *Statistics and Computing*, 28(4):753–774, 2018. URL http://dx.doi.org/10.1007/s11222-017-9761-7. [p34, 35, 36, 38, 39]

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996. [p34]

E. Wit, L. Augugliaro, H. Pazira, J. Gonzalez, and F. Abegaz. Sparse relative risk regression models. *Biostatistics*, 21(2):e131–e147, 2020. URL https://doi.org/10.1093/biostatistics/kxy060. [p34]

Hassan Pazira
Epidemiology and Biostatistics
VU University Medical Center
De Boelelaan 1117, 1081 HV Amsterdam, The Netherlands
h.pazira@amsterdamumc.nl

Luigi Augugliaro
Department of Economics, Business and Statistics
University of Palermo
90128 Palermo, Italy
luigi.augugliaro@unipa.it

Ernst C. Wit
Institute of Computational Science
Università della Svizzera italiana (USI)
Via G. Buffi 13, 6900 Lugano, Switzerland
wite@use.ch

bayesanova: An R package for Bayesian Inference in the Analysis of Variance via Markov Chain Monte Carlo in Gaussian Mixture Models

by Riko Kelter

Abstract This paper introduces the R package **bayesanova**, which performs Bayesian inference in the analysis of variance (ANOVA). Traditional ANOVA based on null hypothesis significance testing (NHST) is prone to overestimating effects and stating effects if none are present. Bayesian ANOVAs developed so far are based on Bayes factors (BF), which also enforce a hypothesis testing stance. Instead, the Bayesian ANOVA implemented in **bayesanova** focusses on effect size estimation and is based on a Gaussian mixture with known allocations, for which full posterior inference for the component parameters is implemented via Markov-Chain-Monte-Carlo (MCMC). Inference for the difference in means, standard deviations and effect sizes between each of the groups is obtained automatically. Estimation of the parameters instead of hypothesis testing is embraced via the region of practical equivalence (ROPE), and helper functions provide checks of the model assumptions and visualization of the results.

1 Introduction

This article introduces **bayesanova**, an R package for conducting a Bayesian analysis of variance (ANOVA) via Markov Chain Monte Carlo (MCMC) in a Gaussian mixture model. Classic frequentist analysis of variance is based on null hypothesis significance testing (NHST), which recently has been shown to produce serious problems regarding the reproducibility and reliability of scientific results (Benjamin et al., 2018; Colquhoun, 2017, 2019; Wasserstein et al., 2019; Wasserstein and Lazar, 2016). NHST is based on test statistics, p-values and significance levels α, which are designed to control the long-term false-positive rate. Still, in a multitude of settings these approaches do in fact lead to an inflated rate of false-positive results, undermining the validity and progress of science. Examples include optional stopping of participant recruiting in studies (Carlin and Louis, 2009) or the necessary testing for violations of distributional assumptions which some frequentist hypothesis tests make (Rochon et al., 2012).

As a solution to these problems, Bayesian methods have been proposed recently and are since gaining popularity in a wide range of scientific domains (McElreath and Smaldino, 2015; Kruschke, 2013, 2015). The Bayesian philosophy proceeds by combining the model likelihood $f(x|\theta)$ with the available prior information $p(\theta)$ to obtain the posterior distribution $f(\theta|x)$ through the use of Bayes' theorem:

$$f(\theta|x) \propto f(x|\theta)f(\theta) \tag{1}$$

While the Bayesian philosophy thus allows for flexible modeling, inference for the posterior distribution $f(\theta|x)$ can be complicated in practice. Therefore, Markov chain Monte Carlo techniques have been developed, which make use of the facts that (1) constructing a Markov chain which has the posterior distribution $f(\theta|x)$ as its stationary distribution, and (2) drawing samples from this Markov chain to approximate the posterior $f(\theta|x)$ can be used to obtain the posterior numerically.

The **bayesanova** package is designed as a Bayesian alternative to the frequentist analysis of variance. By using a Gaussian mixture model and implementing a Markov Chain Monte Carlo algorithm for this model, full posterior inference can be obtained. This allows for explicit hypothesis testing between groups as in the frequentist ANOVA, or for estimation of parameters under uncertainty. The focus in **bayesanova** is on the latter perspective and avoids explicit hypothesis testing. While Bayesian versions of the analysis of variance have been proposed recently by Rouder et al. (2012) and Bergh et al. (2019), these implementations focus on the Bayes factor as a measure of evidence (van Doorn et al., 2019; JASP Team, 2019). As the Bayes factor suffers from multiple problems, one of which is its strong dependence on the used priors – see Kamary et al. (2014) and Robert (2016) – the implementation in **bayesanova** avoids the Bayes factor and uses a different posterior index, the region of practical equivalence (ROPE) (Kruschke, 2018), which has lately been shown to have some desirable properties, in particular in contrast to the Bayes factor (Makowski et al., 2019b).

The plan of the paper is as follows: The next section introduces the analysis of variance in a frequentist and Bayesian fashion and gives an overview about packages implementing these methods.

The following section then introduces the novel approach implemented in **bayesanova**. The details on the mixture representation of the Bayesian analysis of variance are discussed and scenarios where **bayesanova** is designed to be used are detailed. The section thereafter outlines the structure of the package and details the included functions. The following section presents a variety of examples and illustrations using real datasets from biomedical and psychological research as well as synthetic datasets. The last section then provides a summary of the benefits and drawbacks of the used implementation, as well as future plans for the package.

2 Frequentist and Bayesian analysis of variance

Traditional ANOVA models using NHST via the F-statistic

In applied statistics, the one-way analysis of variance is a method which can be used to compare means of two or more samples (typically three). The one-way ANOVA assumes numerical (response) data in each group and (usually) categorical input data like a group indicator in a randomized clinical trial (RCT). Interpreting the ANOVA as a linear model, one obtains for data $y_{i,j}$, where $i = 1, ..., n$ is an index over the experimental units (patients, participants) and $j = 1, ..., k$ an index over treatment groups

$$y_{i,j} = \mu_j + \varepsilon_{i,j} \tag{2}$$

if the experiment is completely randomized. Here, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ so that $\varepsilon_{i,j}$ are normally distributed zero-mean residuals. μ_j is the mean of treatment group j and $y_{i,j}$ the response variable which is measured in the experiment.

The one-way ANOVA then tests the null hypothesis H_0 that all samples are drawn from populations with identical means. To do this, (1) two estimates of the population variance are obtained which rely on various assumptions and (2) an F-statistic is produced by the ANOVA, which is the ratio of variance calculated among the means to the variance within the samples. The intuition here is that if group means are drawn from populations with identical means, the variance of the group means should be smaller than the variance of samples and a high ratio thereby indicates differing means. Mathematical details on computing the F-statistic can be found in the Appendix.

The one-way ANOVA as detailed above makes several assumptions, the most important of which are: (1) variances of populations are equal; (2) responses for a given group are independent and identical distributed random variables; (3) response variable residuals are normally distributed, that is $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

While Monte Carlo studies have shown that the ANOVA is quite robust to small to medium violations of these assumptions (Donaldson, 1966), severe violations of assumptions (1)-(3) will result in inflated rates of false positives and and thereby unreliable results (Tiku, 1971).

Bayesian ANOVA models

Bayesian models for the ANOVA have been developed recently to solve some of the problems of NHST. The developed models can be categorized broadly into two approaches: The first approach relies on the Bayes factor as a measure of relative evidence and was developed by Rouder et al. (2012). The second approach is based on MCMC algorithms like Gibbs sampling in JAGS (Plummer, 2003) or Hamiltonian Monte Carlo (HMC) in Stan (Carpenter et al., 2017; Stan Development Team, 2020). This approach was popularized by Kruschke (2015). Here the region of practical equivalence (ROPE) as introduced by Kruschke (2015) is used for measuring the evidence given the data. Also, an explicit hypothesis testing stance is avoided.

The approach of Rouder et al. (2012) can be summarized as follows: An independent Cauchy prior is considered

$$p(\theta) = \prod_{i=1}^{p} \frac{1}{(1 + \theta_i^2)\pi} \tag{3}$$

for the vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)'$ of the p effects between different groups. For example, in a three-group setting there would be three effects θ_1, θ_2 and θ_3 corresponding to the effects between the first and second, first and third, and second and third group. In the case of $k = 4$ groups, there are $p = 6$ effects and so on. The ANOVA is then rewritten as a linear model

$$\boldsymbol{y} = \boldsymbol{\mu}\mathbf{1} + \sigma \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \tag{4}$$

where $\boldsymbol{\mu}$ is the grand mean parameter, $\mathbf{1}$ a column vector of length n with entries equal to 1, $\boldsymbol{\theta}$ a column

vector of the standardized effect size parameters of length p, and $\boldsymbol{X}$ is the $n \times p$ design matrix. The factor σ in $\sigma\boldsymbol{X}\boldsymbol{\theta}$ is attributed to the reparameterization according to Jeffreys: Following Jeffreys (1961) by reparameterizing $\delta = \mu/\sigma$, where δ is the effect size of Cohen (1988), Rouder et al. (2012) rewrote the observed data sampling distribution as

$$y_i \sim \mathcal{N}(\sigma\delta, \sigma^2) \tag{5}$$

The residuals $\boldsymbol{\varepsilon}$ in Equation (4) are defined to be

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I}) \tag{6}$$

with $\boldsymbol{I}$ being the identity matrix of size n and $\mathbf{0}$ a column vector of zeros of size n.

Putting a Jeffreys prior $p(\mu, \sigma^2) = 1/\sigma^2$ on the mean and variance, and assuming the following g-prior structure

$$\boldsymbol{\theta}|\boldsymbol{G} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{G}) \tag{7}$$

which is based on Zellner (1980), where $\boldsymbol{G}$ is a $p \times p$ diagonal matrix, the only open aspect remaining is putting a prior on the diagonal elements g_l of $\boldsymbol{G}$ for $l = 1, ..., p$. (Rouder et al., 2012) chose

$$g_l \sim \text{Inverse-}\chi^2_1 \tag{8}$$

so that the marginal prior on the effect size parameter vector $\boldsymbol{\theta}$ results in the independent Cauchy distribution given in Equation (3). Rouder et al. (2012) then showed that the resulting BF_{10} can be written as

$$BF_{10} = \int_g K(\boldsymbol{n}, g) \left(\frac{\sum_i \sum_j (y_{ij} - \bar{y})^2 + \frac{1}{g}(\sum_i c_i \bar{y}_i^2 - (\sum_i c_i \bar{y}_i)^2 / (\sum_i c_i))}{\sum_i \sum_j (y_{ij} - \bar{y})^2} \right)^{-(N-1)/2} p(g) dg \tag{9}$$

if a balanced one-way design is used (equal sample sizes in each group). Here, $\boldsymbol{n} = (n_1, ..., n_p)'$ is the vector of sample sizes for each effect $1, ..., p$, $n = \sum_i n_i$ is the full sample size, $c_i = n_i/(n_i + 1/g)$ and

$$K(\boldsymbol{n}, g) = \sqrt{N} \left(\frac{\prod_i 1/(1 + g n_i)}{\sum_i n_i/(1 + g n_i)} \right)^{1/2} \tag{10}$$

In summary, this Bayes factor of Rouder et al. (2012) can be computed via Gaussian quadrature, as it constitutes a one-dimensional integral after inserting the necessary quantities.

The second approach of a Bayesian ANOVA model can be credited to Kruschke (2015), who uses the MCMC sampler JAGS (Plummer, 2003) to obtain full posterior inference in his model instead of relying on the Bayes factor. The reasons for avoiding the Bayes factor as a measure of evidence are that (1) it depends strongly on the selected prior modeling (Kamary et al., 2014); (2) the Bayes factor states only relative evidence for the alternative to the null hypothesis (or vice versa) so that even a large Bayes factor does not indicate that either one of both hypotheses is a good fit for the actual data (Kelter, 2020a,b); (3) it can be located in the same formalism of hypothesis testing the pioneers of frequentist testing advocated at the time of invention (Robert, 2016; Tendeiro and Kiers, 2019). In addition, the calculation of the Bayes factor for increasingly complex models can be difficult, as the above derivations of Rouder et al. (2012) exemplify, see also Kamary et al. (2014). Importantly, the Bayes factor assigns positive measure to a Lebesgue-null-set which is puzzling from a measure-theoretic perspective, compare Kelter (2021c), Rao and Lovric (2016), and Berger (1985).

Kruschke (2015) modeled the Bayesian ANOVA for k groups and n observations $y_1, ...y_n$ as a hierarchical Bayesian model, where

$$y_i \sim \mathcal{N}(\mu, \sigma_y^2) \tag{11}$$

where the standard deviation σ_y is modelled as

$$\sigma_y \sim \mathcal{U}(L, H) \tag{12}$$

the mean μ_i is the linear combination

$$\mu = \beta_0 + \sum_{j=1}^{k} \beta_j x_j(i) \tag{13}$$

and the coefficients of this linear combination are given as

$$\beta_0 \sim \mathcal{N}(M_0, S_0) \tag{14}$$
$$\beta_j \sim \mathcal{N}(0, \sigma_\beta) \tag{15}$$

where $x_j(i)$ is the index for the group the observation y_i belongs to. If, for example, y_i is in the first group, $x_1(i) = 1$ and $x_j(i) = 0$ for all $j \neq 1$ with $j \in \{1, ..., k\}$, yielding the group mean $\mu_i = \beta_0 + \beta_1$ of the first group. Thus, although Equation (11) seems to indicate that there is a single mean μ for all observations y_i, $i = 1, ..., n$, the mean μ takes k different values depending on which group the observation y_i is located in. These k different values for μ correspond to the different means in the k groups as shown in Equation (13). The variables L, H, M_0, S_0 are hyperparameters, and the parameter β_j can be interpreted as the effect size differing from the grand mean β_0, which is why the prior on β_j is normal with mean zero so that the expectation of these effect size differences from the grand mean sum up to zero again. The hyperparameter σ_β can either be set constant or given another prior, extending the multilevel model, where Kruschke (2015) followed the recommendations of Gelman and Hill (2006) to use a folded t-distribution or a gamma-distribution with non-zero mode.

Inference for the full posterior, that is for the parameters $\mu_k, \sigma_y, \beta_0, \beta_j \forall j, j = 1, ..., k$ (and σ_β, if a hyperprior like a folded t-distribution or gamma-distribution is used on this parameter) given the data is provided via the MCMC sampler JAGS (Plummer, 2003), which uses Gibbs sampling to draw samples from the posterior. Posterior distributions obtained through Gibbs sampling are finally used to estimate all parameters via 95% Highest-Density-Intervals (HDI). Explicit testing is avoided.

3 Available software

Available software for the traditional ANOVA

Conducting a traditional analysis of variance is possible with an abundance of software, for example via the **stats** package (R Core Team, 2020) which is part of the R programming language (R Core Team, 2020).

Available software for the Bayesian ANOVA

The **BayesFactor** package by Morey and Rouder (2018) provides the Bayesian ANOVA Bayes factor of Rouder et al. (2012), and various helper functions for analysis of the results.

A simple illustration of the main workflow in the **BayesFactor** package is given here, using the `ToothGrowth` dataset in the **datasets** package (Cannon et al., 2019). The `ToothGrowth` dataset contains three columns: `len`, the dependent variable each of which is the length of a guinea pig's tooth after treatment with vitamin C. The predictor `supp` corresponds to the supplement type (either orange juice or ascorbic acid), the predictor `dose` is the amount of vitamin C administered.

The **BayesFactor** package's core function allows the comparison of models $\mathcal{M}_0, ..., \mathcal{M}_n$ with factors as predictors. The null model without any predictors is most often compared to models including predictors or even interaction terms using the Bayes factor as detailed above. The function `anovaBF` computes several model estimates at once, so that the model with the largest Bayes factor can be selected. The data are first loaded and the categorial predictors converted to factors:

```
R> set.seed(42)
R> library(datasets)
R> data(ToothGrowth)
R> head(ToothGrowth,n=3)

   len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5

R> ToothGrowth$dose = factor(ToothGrowth$dose)
R> levels(ToothGrowth$dose) = c('Low', 'Medium', 'High')
```

Then, a Bayesian ANOVA is conducted using both predictors `dose`, `supp` and the interaction term `dose * supp`:

```
R> library(BayesFactor)
R> bf = anovaBF(len ~ supp * dose, data = ToothGrowth)

Bayes factor analysis
--------------
[1] supp                  : 1.198757      +- 0.01%
[2] dose                  : 4.983636e+12  +- 0%
[3] supp + dose           : 2.963312e+14  +- 1.59%
[4] supp + dose + supp:dose : 8.067205e+14  +- 1.94%

Against denominator:
  Intercept only
---
Bayes factor type: BFlinearModel, JZS
```

The results are shown in form of the Jeffreys-Zellner-Siow (JZS) Bayes factor BF_{10} detailed previously. As the BF_{10} for the model including both predictors `supp` and `dose` is largest, the Bayesian ANOVA favours this model over the null model which includes only the intercept. Thus, as there are the low, medium and high dose groups and the two supplement groups, in total one obtains $3 \times 2 = 6$ different groups. The results show that there is strong evidence that the model attesting these six differing groups is favourable over the null model (and every other model as given in output lines `[1]`, `[2]` and `[3]`).

Note, that this solution is also implemented in the open-source software JASP, for an introduction see Bergh et al. (2019).

The Bayesian ANOVA model of Kruschke (2015) is not implemented in a software package by now. Instead, users have to write their own model scripts for JAGS (Plummer, 2003) to run the analysis. Still, recently the package **BANOVA** was published by Dong and Wedel (2019), which uses JAGS (Plummer, 2003) and the Hamiltonian Monte Carlo (HMC) sampler Stan (Carpenter et al., 2017) via the package **RStan** (Stan Development Team, 2020) to provide similar inferences without the need to code the JAGS or Stan models on your own.

Note that in the above example, a traditional ANOVA can easily be fit via

```
R> summary(aov(len ~ supp * dose, data = ToothGrowth))

            Df Sum Sq Mean Sq F value   Pr(>F)
supp         1  205.4   205.4  15.572 0.000231 ***
dose         2 2426.4  1213.2  92.000  < 2e-16 ***
supp:dose    2  108.3    54.2   4.107 0.021860 *
Residuals   54  712.1    13.2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

which yields similar results, favouring the full model with both predictors and interaction term, as both predictors and the interaction term are significant.

4 The Bayesian ANOVA model based on Gaussian mixtures

The method used in the **bayesanova** package is based on estimation of parameters in a Gaussian mixture distribution. On this mixture a Gibbs sampling algorithm is applied to produce posterior distributions of all unknown parameters given the data in the Gaussian components, that is for $\mu_j, \sigma_j, j = 1, ..., k$ and for the differences in means $\mu_l - \mu_r, l \neq r$ and the effect sizes $\delta_{lr}, l \neq r$ where k is the number of groups in the study or experiment. This way, a relatively complete picture of the situation at hand can be drawn and while the technical aspects are omitted here, the validity of the procedure stems from standard MCMC theory, see for example Robert and Casella (2004). The principal idea of mixture models is expressed by Frühwirth-Schnatter (2006):

> *Consider a population made up of K subgroups, mixed at random in proportion to the relative group sizes $\eta_1, ..., \eta_K$. Assume interest lies in some random feature Y which is heterogeneous across and homogeneous within the subgroups. Due to heterogeneity, Y has a different probability distribution in each group, usually assumed to arise from the same parametric family $p(y|\theta)$ however, with the parameter θ differing across the groups. The groups may be labeled through a discrete indicator variable S taking values in the set $\{1, ..., K\}$.*
> *When sampling randomly from such a population, we may record not only Y, but also the*

> *group indicator S. The probability of sampling from the group labeled S is equal to η_S, whereas conditional on knowing S, Y is a random variable following the distribution $p(y|\theta_S)$ with θ_S being the parameter in group S. (...) The marginal density $p(y)$ is obviously given by the following mixture density*

$$p(y) = \sum_{S=1}^{K} p(y, S) = \eta_1 p(y|\theta_1) + ... + \eta_S p(y|\theta_K)$$

Clearly, this resembles the situation of the analysis of variance, in which the allocations S are known. Traditionally, mixtures are treated with missing allocations but in the setting of the ANOVA these are known, leading to a much simpler scenario. This interpretation also makes sense from a semantic point: the inherent assumption of a researcher is that the population is indeed made up of k subgroups in the case of a k-group ANOVA, which differ in a random feature Y which is heterogeneous across groups and homogeneous within each group. When conducting for example a randomized clinical trial (RCT), the group indicator S is of course recorded. The clinician will choose the patients according to a sampling plan, which could be designed to achieve equally sized groups, that is, $\eta_1 = \eta_2 = ... = \eta_k$ for k study groups. Thus, when sampling the population with the target of equally sized groups, the researcher will sample the objects with equal probability from the population. Consider a treatment one, treatment two and a control group. In this typical setting, the researcher could flip a coin for each patient in the RCT to assign him or her to one of the two treatment groups or to the control group, so that with probability $\eta_1 = \eta_2 = \eta_3 = 1/3$ for any group, the patient is assigned to it. Repeating this process then leads to the mixture model given above. After the RCT is conducted, the resulting histogram of observed Y values will finally take the form of the mixture density $p(y)$ above. If there is an effect in the treatment, this density $p(y)$ will express three modes which in turn result from the underlying mixture model of the data-generating process.

If unbalanced groups are the goal, weights can be adjusted accordingly, for example $\eta_1 = 0.3$, $\eta_2 = 0.2$ and $\eta_3 = 0.5$. After fixing the mixture weights η_1, η_2, η_3, the family of distributions for the mixture components needs to be selected. The above considerations lead to finite mixtures of normal distributions which *'occur frequently in many areas of applied statistics such as [...] medicine'* (Frühwirth-Schnatter, 2006, p. 169). The components $p(y|\theta_i)$ therefore become $f_N(y; \mu_j, \sigma_j^2)$ for $j = 1, ..., k$ in this case, where $f_N(y; \mu_j, \sigma_j^2)$ is the density of the univariate normal distribution. Parameter estimation in finite mixtures of normal distributions consists of estimation of the component parameters (μ_j, σ_j^2), the allocations $S_i, i = 1, ..., n$ and the weight distribution $(\eta_1, ..., \eta_k)$ based on the available complete data $(y_i, S_i), i = 1, ..., n$. In the case of the Bayesian ANOVA, the allocations S_i (where $S_i = 1$ if y_i belongs to the first component, $S_i = 2$ if y_i belongs to the second component, until $S_i = k$ if y_i belongs to the k-th group) are known for all observations y_i, $i = 1, ..., n$. Therefore, inference reduces to inference for the density parameters (μ_j, σ_j^2) of the normal components of the mixture for the $j = 1, ..., k$ groups.

The Bayesian ANOVA model based on Gaussian mixtures is summarized in Figure 1 using the three-group case as an example:

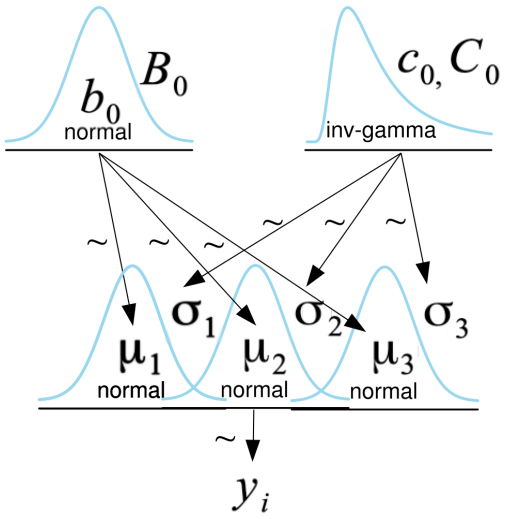


Figure 1: Three-component Gaussian mixture with known allocations for the Bayesian analysis of variance

The measured variables y_i follow a three-component Gaussian mixture with known allocations.

Model	Purpose	Evidence measure	Computational aspects
Frequentist ANOVA	Testing the global hypothesis that all samples are drawn from populations with identical means against the alternative	F-statistic and p-value	Analytic solution
Bayesian ANOVA of Rouder et al. (2012)	Test the global hypothesis that the effect size vector is zero versus the alternative	Bayes factor	Numerical integration required
Bayesian ANOVA of Kruschke (2015)	Estimation of effect sizes between groups via ROPE and 95% HPD	ROPE	Gibbs sampling in MCMC sampler JAGS (or Stan) required
Bayesian ANOVA based on Gaussian mixtures	Estimation of effect sizes between groups via the ROPE and posterior probability mass	ROPE	Gibbs sampling without MCMC sampler JAGS (or Stan) required

Table 1: Overview about the four ANOVA models

The first group is normally distributed as $\mathcal{N}(\mu_1, \sigma_1)$, the second group as $\mathcal{N}(\mu_2, \sigma_2)$ and the third group as $\mathcal{N}(\mu_3, \sigma_3)$. The means μ_1, μ_2 and μ_3 are each distributed as $\mu_j \sim \mathcal{N}(b_0, B_0), j = 1, 2, 3$ with noninformative hyperparameters b_0 and B_0 and the standard deviations σ_1, σ_2 and σ_3 are distributed as $\sigma_j \sim \mathcal{G}^{-1}(c_0, C_0), j = 1, 2, 3$ with noninformative hyperparameters c_0 and C_0. For details, see Kelter (2021d, 2020c). As the allocations are known, the weights η_1, η_2 and η_3 are known too, and need not to be estimated, which is why the parameters η_1, η_2, η_3 are not included in the diagram. The model visualized in Figure 1 can be generalized for an arbitrary number of mixture components, which then includes nearly arbitrary ANOVA settings for comparison of multiple groups. A definitive advantage of this model is that inference is obtained for both means and standard deviations, yielding richer information compared to the testing perspectives which are stressed in traditional or Bayesian ANOVA models focussing on the Bayes factor. Also, posterior distributions of effect sizes can be obtained via MCMC, providing an additional layer of information to draw inferences.

Instead of relying on the Bayes factor, the **bayesanova** package follows the approach of Kruschke (2018) to use a region of practical equivalence (ROPE). The effect size δ is routinely categorized as small, medium or large in medical research when $\delta \in [0.2, 0.5), \delta \in [0.5, 0.8)$ or $\delta \in [0.8, \infty)$, see Cohen (1988). The approach using the ROPE proceeds by taking these categories as regions of practical equivalence, that is both $\delta = 0.25$ and $\delta = 0.26$ are identified as a small effect because both are inside the region of practical equivalence $[0.2, 0.5)$ of a small effect δ. The underlying idea is that measuring effect sizes only makes sense up to a specific precision, which is given by the above categorization of effect sizes. By studying how much probability mass of the posterior distribution of δ lies inside some of the above ROPEs $[0.2, 0.5)$, $[0.5, 0.8)$ and $[0.8, \infty)$ of a small, medium and large positive effect for δ (negative effects analogue), a continuous statement about the most probable effect size δ given the data can be made. Kruschke originally advocated to use the location of the 95% highest-posterior-density (HPD) interval in relation to the ROPE to test whether the null value in the middle of the ROPE should be accepted or rejected for practical purposes. Here, this approach is generalized by switching to the amount of posterior probability mass inside the ROPE. Detailed examples are provided later in this paper.

Table 1 provides an overview about the four ANOVA models and their purpose. Although it appears that the model of Kruschke (2015) and the Gaussian mixture modeling approach proposed in this paper have the same purpose, they differ in how data y_i are assumed to be generated. In the mixture approach we assume that the sample of n_j participants in group j results from a mixture process, e.g. by flipping a coin, rolling a dice or using any other randomization device (as is the case in clinical trials when assigning patients to groups according to a double-blinded protocol). Thus, the process of data generation is not "one has collected n_j participants for group j" but "the given sample of n_j participants in group j is assumed to be a realization of a mixture process where with probability η_j participants are assigned to group j". Importantly, note that the realization of n_j participants in group j for $j = 1, ..., k$ is expected under the mixture component weight $\eta_j = n_j/n$, but also entirely different group sizes n_j can result under such a mixture. In fact, the weights $\eta_j = n_j/n$ which are assumed to be known are the corresponding maximum-likelihood-estimators of the weight parameters η_j given the sample sizes n_j for $j = 1, ..., k$, but the conceptual focus of the mixture approach is to

Function	Description
`bayes.anova`	Main function of the package, conducts the MCMC algorithm to provide full posterior inference in the three-component Gaussian mixture model
`assumption.check`	Helper function for checking the assumption of normality in each group previous to running a Bayesian ANOVA
`anovaplot`	Provides multiple visualizations of the results, including posterior distributions, difference in means and standard deviations and effect sizes as well as a full ROPE-analysis
`post.pred.check`	Provides a posterior predictive check for a fitted Bayesian ANOVA model

Table 2: Outline of the four main functions implemented in **bayesanova**

closely mimic the randomization process researchers follow when conducting a randomized controlled trial. Note further that the model of Kruschke assumes homogeneity of variances in contrast to the Gaussian mixture model, but Kruschke's model can easily be extended to account for heterogeneity of variance, rendering this difference less important. Note that both the frequentist ANOVA and the Bayesian version of Rouder et al. (2012) assume homogeneity of variance across groups.

5 Package structure and implementation

The **bayesanova** package has four functions. These provide (1) the MCMC algorithm for conducting the Bayesian ANOVA in the Gaussian mixture model with known allocations, detailed above, (2) checks of the model assumptions and (3) visualizations of the posterior results for easy interpretation and communication of research results. Visualizations of the posterior mixture components in comparison with the original data are provided by the fourth function. An overview is provided in Table 2.

The core function is `bayes.anova`, which provides the MCMC algorithm to obtain full posterior inference in a k-component Gaussian mixture model shown in Figure 1 for the special case of $k = 3$ components. The function implements a Gibbs sampling algorithm, which iteratively updates

1. the means $\mu_j|\mu_{-j}, \sigma_1, ..., \sigma_k, S, y$ given the other means μ_{-j} and standard deviations $\sigma_1, ..., \sigma_k$ as well as the full data S, y, where S is the indicator vector for the groups the observations y belong to
2. the standard deviations $\sigma_j|\sigma_{-j}, \mu_1, ..., \mu_k, S, y$ given the other standard deviations σ_{-j} and means $\mu_1, ..., \mu_k$ as well as the full data S, y, where S is again the indicator vector for the groups the observations y belong to

The details of the Gibbs sampler can be found in Kelter (2020c, 2021d), and the validity of the method follows from standard MCMC theory, see for example Robert and Casella (2004).

The `bayes.anova` function takes as input three numerical vectors `first`, `second` and `third`, which correspond to the observed responses in each of the three groups and provides multiple optional parameters:

```
bayes.anova(n=10000, first, second, third,
fourth = NULL, fifth = NULL, sixth = NULL,
hyperpars="custom", burnin=n/2, sd="sd", q=0.1, ci=0.95)
```

These are the only mandatory input values, and currently six groups are the limit **bayesanova** supports. More than three groups can be handed to the function by providing numerical vectors for the parameters `fourth`, `fifth` and `sixth`.

If no other parameters are provided, the function chooses a default of `n=10000` Gibbs sampling iterations, where the burn-in of the Markov chains is set to `burnin=n/2`, so that the first 5000 iterations are discarded. The default setting uses inference for means μ_j and standard deviations σ_j, which is indicated by the parameter `sd="sd"`, but inference for variances σ_j^2 instead of standard deviations σ_j can easily be obtained by setting `sd="var"`. The credible level for all computed credible intervals defaults to 0.95, indicated by `ci=0.95`. The two remaining parameters `hyperpars` and `q` define preselected values for the hyperparameters in the prior, to ensure weakly informative priors are used which influence the analysis as little as possible. For details, see Kelter (2020c, 2021d), but in general these values apply to a broad range of contexts so that changing them is not recommended. Note, that another set of hyperparameters based on Raftery (1996) can be selected via `hyperpars="rafterys"`, if desired.

After execution, the function returns a dataframe including four Markov chains for each parameter of the specified size `n-burnin`, to make subsequent convergence assessment or post-processing of the MCMC results possible.

The second function is `assumption.check`. This function runs a preliminary assumption check on the data, which is recommended before running a Bayesian ANOVA. The model assumptions are normality in each mixture component, so that the `assumption.check` function runs Shapiro-Wilk tests to check for normality (Shapiro and Wilk, 1965). The input parameters are the three numerical vectors `x1`, `x2` and `x3` including the observed responses in the first, second and third group, and the desired confidence level `conf.level` for the Shapiro-Wilk tests:

```
assumption.check(x1, x2, x3, x4 = NULL, x5 = NULL, x6 = NULL, conf.level=0.95)
```

The default confidence level is `0.95`. More than three groups can easily be added by providing values for `x4`, `x5` and `x6`.

The third function is `anovaplot`, which provides a variety of visualizations of results. The function takes as input a dataframe `dataframe`, which should be the result of the `bayes.anova` function detailed above, a parameter `type`, which indicates which visualization is desired, a parameter `sd`, which indicates if the provided dataframe includes posterior draws of σ_j or σ_j^2 and last a parameter `ci`, which again defined the credible level used in the computations.

```
anovaplot(dataframe, type="rope", sd="sd", ci=0.95)
```

The default values for `sd` is `"sd"`, and the default credible level is `0.95`. The `type` parameter takes one of four possible values: (1) `type="pars"`, (2) `type="diff"`, (3) `type="effect"` and (4) `type="rope"`. In the first case, posterior distributions of all model parameters are produced, complemented by convergence diagnostics in form of trace plots, autocorrelation plots and the Gelman-Brooks-Rubin shrink factor (Gelman and Brooks, 1998), which should be close to one to indicate convergence to the posterior. In the second case, the posterior distributions of the differences $\mu_i - \mu_j, j \neq i$ of the group means and differences $\sigma_l - \sigma_r, l \neq r$ of the group standard deviations (or variances, if `sd="var"` and the dataframe includes posterior draws of the σ_j^2's instead of σ_j's) are produced, complemented by the same convergence diagnostics. In the third case, the posterior distributions of the effect sizes $\delta_{lr}, l \neq r$ are produced, which are most often of interest in applied research. In this case, posteriors are complemented by the same convergence diagnostics, too. The last and fourth case produces a full ROPE-analysis, which does provide the posteriors of the effect sizes $\delta_{lr}, l \neq r$, but additionally computes a partitioning of the posterior probability mass into the standardized ROPEs of small, medium and large (and no) effect sizes according to Cohen (1988), which are the reference standard in medical and psychological research.

The last function `post.pred.check` provides a posterior predictive check for a fitted Bayesian ANOVA model against the original data, which is routine in a Bayesian workflow Gabry et al. (2019).

6 Illustrations and examples

This section provides illustrations and a variety of examples, in which the **bayesanova** package can be used and provides richer information than existing solutions.

Tooth growth of guinea pigs treated with vitamin C

The guinea pig dataset from above is used as a first example. The data are included in the dataset `ToothGrowth` in the **datasets** package which is part of R. First, data is loaded and split into three groups, corresponding to a low, medium and high administered vitamin C dose:

```
R> library(datasets)
R> data(ToothGrowth)
R> head(ToothGrowth,n=3)

   len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5

R> library(dplyr)
R> library(tidyr)
R> library(bayesanova)
```

```
R> grp1 = (ToothGrowth %>% filter(dose==0.5) %>% select(len))$len
R> grp2 = (ToothGrowth %>% filter(dose==1.0) %>% select(len))$len
R> grp3 = (ToothGrowth %>% filter(dose==2.0) %>% select(len))$len
```

Next, we run the assumption checks on the data

```
R> assumption.check(grp1, grp2, grp3, conf.level=0.95)

Model assumptions checked. No significant deviations from normality detected.
Bayesian ANOVA can be run safely.
```

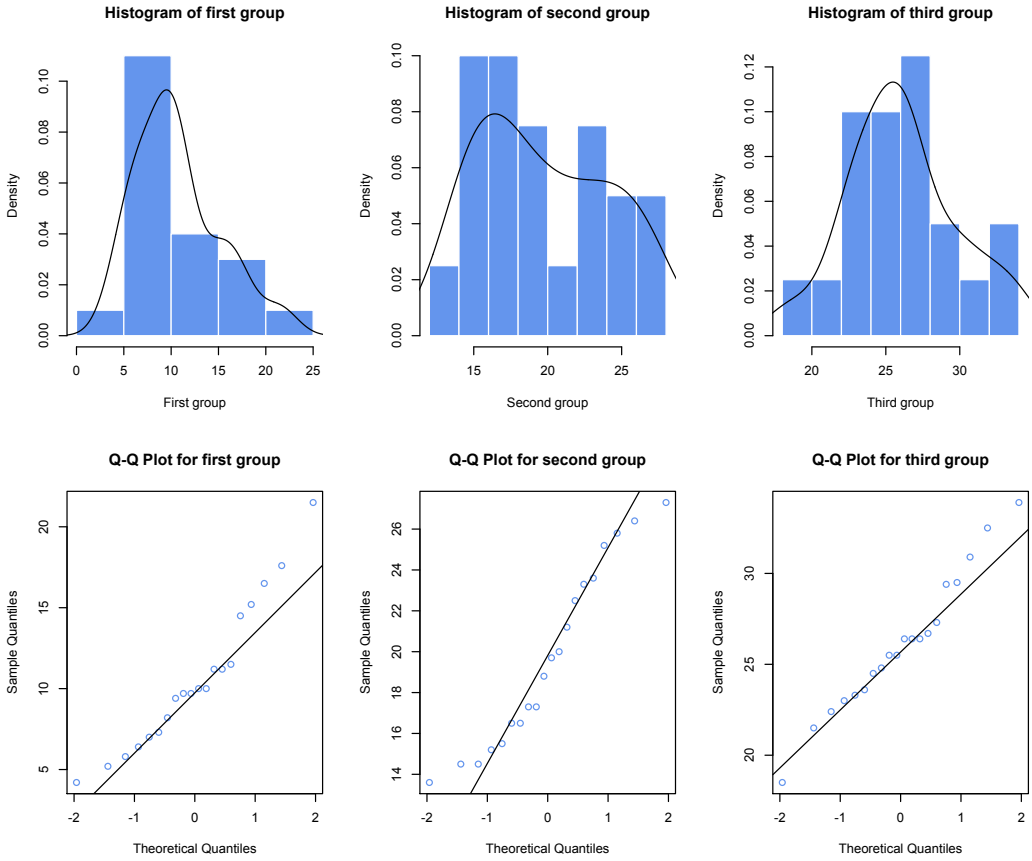


Figure 2: Assumption checks for the `ToothGrowth` dataset using the `assumption.check()` function in **bayesanova**, showing that data in the three groups can be assumed as normally distributed so that running the Bayesian ANOVA based on the Gaussian mixture model is justified

Figure 2 shows the histograms and quantile-quantile plots for all three groups produced by `assumption.check()`. Clearly, there are no large deviations, and no Shapiro-Wilk test was significant at the 0.05 level.

Next, the Bayesian ANOVA can be run via the `bayes.anova` function. Therefore, the default parameter values are used, yielding `n=5000` posterior draws:

```
R> set.seed(42)
R> res = bayes.anova(first = grp1, second = grp2, third = grp3)
```

```
|Parameter     |LQ    |Mean  |UQ    |Std.Err |
|:-------------|:-----|:-----|:-----|:-------|
|mu1           |8.69  |10.61 |12.5  |0.91    |
|mu2           |18.05 |19.75 |21.46 |0.84    |
|mu3           |24.94 |26.1  |27.25 |0.57    |
|sigma1        |3.02  |4.07  |5.67  |0.67    |
|sigma2        |2.95  |3.96  |5.43  |0.64    |
|sigma3        |2.43  |3.25  |4.42  |0.52    |
|mu2-mu1       |6.7   |9.15  |11.7  |1.25    |
|mu3-mu1       |13.42 |15.49 |17.67 |1.06    |
```

```
|mu3-mu2       |4.36  |6.34  |8.38  |1.01    |
|sigma2-sigma1 |-2.02 |-0.11 |1.68  |0.93    |
|sigma3-sigma1 |-2.62 |-0.81 |0.74  |0.85    |
|sigma3-sigma2 |-2.46 |-0.71 |0.85  |0.82    |
|delta12       |-5.77 |-4.59 |-3.21 |0.65    |
|delta13       |-9.37 |-8.14 |-6.63 |0.71    |
|delta23       |-4.36 |-3.36 |-2.19 |0.56    |
```

The results table shows the lower and upper quantile, corresponding to the $100 \cdot \texttt{ci} + (100 - \texttt{ci})/2$ and $(100 - \texttt{ci})/2$ quantiles where `ci` is the credible level chosen above. Also, the posterior mean and standard error are given for each parameter, difference of parameters and effect size. The results clearly show that there are huge differences between the groups: For example, one can immediately spot that the more vitamin c given, the more tooth growth can be observed via tooth lengths. While the first group (low dose) has a posterior mean of 10.61 with credible interval $[8.69, 10.61]$, the second group achieves a mean of 19.75 with credible interval $[18.05, 21.46]$. The third group has a posterior mean of even 26.1 with credible level [24.94.27.25]. The posterior estimates for the differences $\mu_2 - \mu_1$, $\mu_3 - \mu_1$ and $\mu_3 - \mu_2$ show that all groups differ from each other with a very high probability, given the data.

Note that the information provided is much more fine-grained than in the solutions via the traditional ANOVA and the Jeffreys-Zellner-Siow based Bayes-factor ANOVA above. While in these two solutions, one could only infer that the model using both predictors and the interaction term is the best, now we are given precise estimates of the effect sizes between each group defined by the dose of vitamin c administered. Note also, that including the second predictor `supp` is no problem, leading to a setting which incorporates six groups in the mixture then.

Heart rate data for runners

The second example is from the biomedical sciences and uses the heart rate data from Moore et al. (2012). In the study, heart rates of female and male runners and generally sedentary participants (not regularly running) following six minutes of exercise were recorded. The participant's `Gender` and `Heart.rate` are given and which group he or she belongs to (`Group=="Runners"` or `Group=="Control"`). In the study, 800 participants were recruited, so that in each of the four groups given by the combinations of `Gender` and `Group` 200 subjects participated.

Therefore, the situation requires a 2×2 between subjects ANOVA. Specifically, interest lies in the hypothesis that heart rate differs between gender and groups. The Bayesian ANOVA of **bayesanova** can easily be applied in such an often encountered setting. We first load the data and split them into the four groups:

```
R> library(dplyr)
R> hr=read.csv("heartrate.csv",sep=",")
R> head(hr)

  Gender   Group Heart.Rate
1 Female Runners        119
2 Female Runners         84
3 Female Runners         89
4 Female Runners        119
5 Female Runners        127
6 Female Runners        111

R> femaleRunners = (hr %>% filter(Gender=="Female")
+  %>% filter(Group=="Runners")
+  %>% select(Heart.Rate))$Heart.Rate
R> maleRunners = (hr %>% filter(Gender=="Male") %>% filter(Group=="Runners")
+  %>% select(Heart.Rate))$Heart.Rate
R> femaleControl = (hr %>% filter(Gender=="Female")
+  %>% filter(Group=="Control")
+  %>% select(Heart.Rate))$Heart.Rate
R> maleControl = (hr %>% filter(Gender=="Male") %>% filter(Group=="Control")
+  %>% select(Heart.Rate))$Heart.Rate
```

Then, we check the model assumptions:

```
R> assumption.check(femaleRunners, maleRunners, femaleControl, maleControl)
```

We can thus safely proceed running the Bayesian ANOVA:

```
R> set.seed(42)
R> resRunners = bayes.anova(first = femaleRunners, second = maleRunners,
+  third = femaleControl, fourth = maleControl)
```

Parameter	LQ	Mean	UQ	Std.Err
mu1	113.48	116	118.5	1.27
mu2	102.51	103.98	105.55	0.76
mu3	145.44	148.04	150.52	1.3
mu4	127.12	130.01	132.82	1.47
sigma1	14.38	15.87	17.51	0.8
sigma2	11.21	12.35	13.67	0.63
sigma3	14.71	16.19	17.85	0.82
sigma4	15.46	17.02	18.79	0.85
mu2-mu1	-14.9	-12.01	-9.06	1.48
mu3-mu1	28.47	32.04	35.6	1.83
mu4-mu1	10.19	14.01	17.9	1.96
mu3-mu2	41.12	44.05	46.95	1.51
mu4-mu2	22.83	26.02	29.21	1.66
mu4-mu3	-21.8	-18.03	-14.4	1.92
sigma2-sigma1	-5.6	-3.52	-1.57	1.02
sigma3-sigma1	-1.94	0.32	2.53	1.15
sigma4-sigma1	-1.14	1.15	3.51	1.18
sigma3-sigma2	1.83	3.84	5.85	1.03
sigma4-sigma2	2.7	4.67	6.8	1.05
sigma4-sigma3	-1.48	0.83	3.13	1.17
delta12	2.4	3.2	3.96	0.4
delta13	-8.92	-8.01	-7.05	0.48
delta14	-4.42	-3.46	-2.5	0.49
delta23	-12.55	-11.67	-10.77	0.45
delta24	-7.65	-6.79	-5.91	0.45
delta34	3.52	4.43	5.37	0.48

The results reveal multiple insights now. To support the interpretation, we first produce visualisations of the results via the `anovaplot()` function:

```
R> anovaplot(resRunners)
```

Figure 3 shows the plots produces by the above call to `anovaplot()`. The first row shows the posterior distributions of the effect sizes δ_{12}, δ_{13} and δ_{23}. The second row below is the analysis based on the ROPE, which partitions the posterior probability mass into the standard ROPES for effect sizes according to Cohen (1988).

Thus, we can see that for δ_{12} – which equals the effect size between female runners and male runners – there is a very large effect with posterior mean 3.2 and 95% credible interval $[2.402, 3.96]$, confirmed by the fact that 100% of the posterior probability mass are located inside the ROPE of a large effect according to Cohen (1988) (which includes values ≥ 0.8). Based on the results, the posterior probability of a large effect between female and male runners given the data is one, which means female runners have a faster heart beat after exercising six minutes than male runners.

To check if this effect exists also in the control groups, we compare the posterior of δ_{34}, corresponding to the effect size between the female and male controls. The results are given in the right plot of the third and fourth row in 3 and show that also in the control groups the effect is present. Here, the effect size is estimated to be even larger than for the runner groups with a posterior mean of 4.427 and a 95% credible interval $[3.517, 5.366]$. Thus, regular running seems to reduce the observed heartbeat differences between males and females in the form of a large effect. We could proceed this way and compare all other groups, too.

To check the model fit, we use the `post.pred.check` function, which performs a posterior predictive check against the observed data by drawing `reps` samples from the posterior distribution and visualizing the original data's density with density overlays for the reps sampled posterior predictive densities of the data:

```
post.pred.check(anovafit = resRunners, ngroups = 4, out = hr$Heart.Rate ,
reps = 50, eta = c(1/4,1/4,1/4,1/4))
```

The argument `anovafit` takes the resulting dataframe of the `bayes.anova` function as input, the

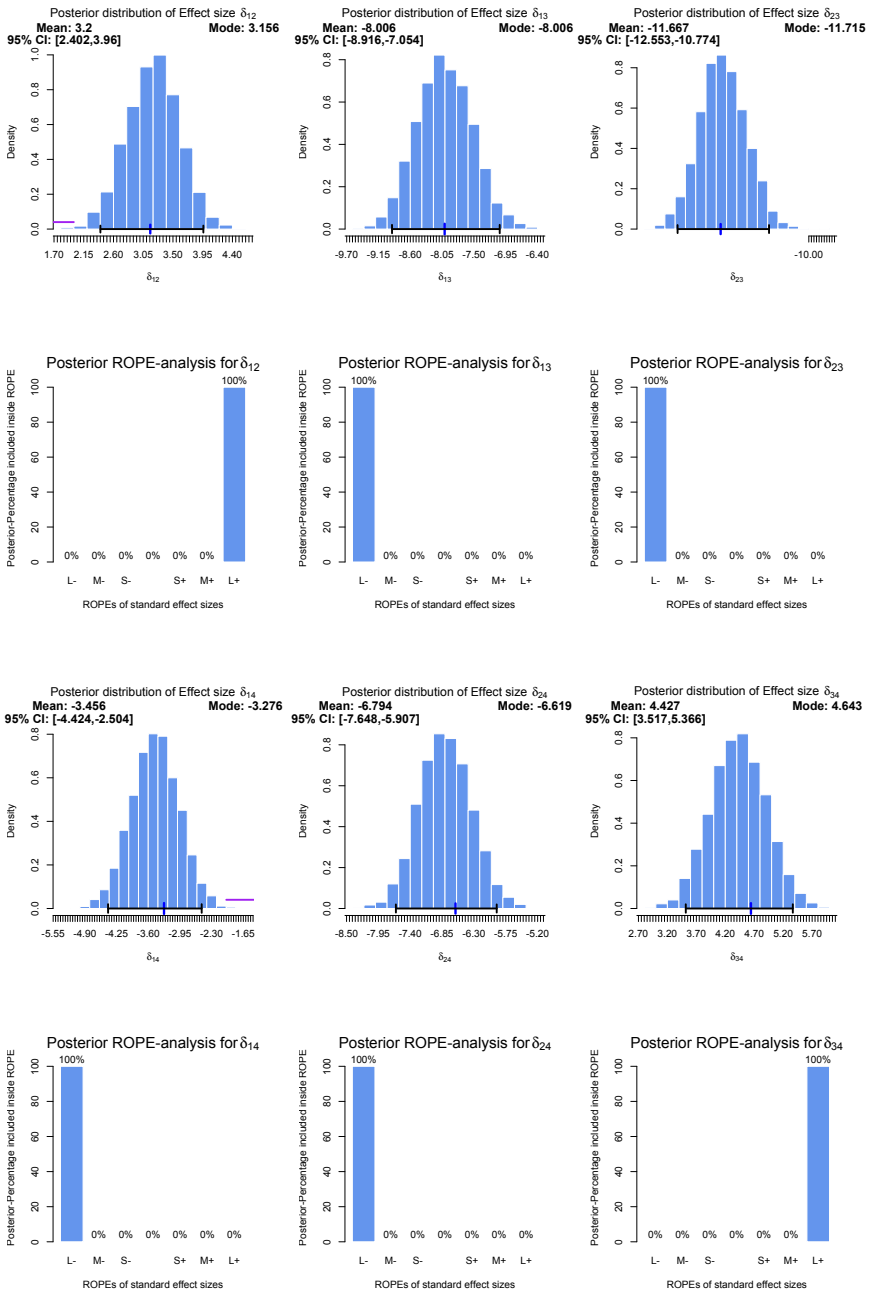


Figure 3: Visualisations of the results for the `Heart rate` dataset using the `anovaplot()` function in **bayesanova**, showing (1) the resulting posterior distributions of the effect sizes between each pair of groups (first and third row) and (2) the posterior ROPE-analysis for each group comparison (second and fourth row)

number of groups is specified in `ngroups`, `out` is the vector of all data originally observed (no matter which group), `reps` is the number of posterior predictive density overlays desired, and `eta` is the vector of weights used in the Gaussian mixture. Here, as all four groups include 200 participants, each weight is 1/4. The resulting posterior predictive check is shown in the left plot of 4, and indicates that while there is some overdispersion in the center of the posterior predictive distributions simulated, the overall fit seems reasonable.

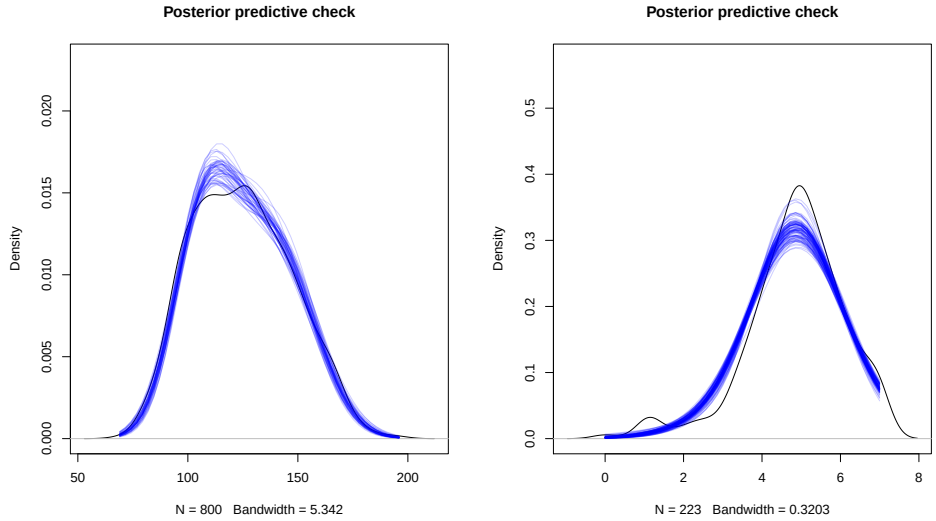


Figure 4: Posterior predictive checks using the `post.pred.check` function; left: For the `runners` dataset; right: For the `feelings` dataset; in both cases, results show that the overall fit of the Gaussian mixture model is reasonable

Pleasantness ratings after watching artistic or nude pictures

This example uses data from a study conducted by Balzarini et al. (2017), in which men and women's feelings towards their partners after watching either erotic or abstract art pictures were analysed. The study was published in the *Journal of Experimental Social Psychology*, and also the average pleasantness obtained from viewing the pictures was studied, as one of the research questions was whether men and women rate pleasantness of the pictures differently for nude and abstract art. This leads to a 2×2 factorial ANOVA for the variables gender and picture type, coded as `Gender` and `Condition` in the dataframe.

First, data is loaded and split into the four groups of interest:

```
R> feelings=read.csv("feelings.csv",sep=",")
R> head(feelings)

  Gender Age  RelLen    Condition PartnerAttractiveness
1   Male  43  3.7500        Nudes                    21
2 Female  26  3.0000        Nudes                    19
3 Female  35  5.2500 Abstract Art                    27
4 Female  31  2.0000 Abstract Art                    22
5 Female  23  4.0000 Abstract Art                    27
6   Male  36 19.9167        Nudes                    16
  LoveForPartner AveragePleasantness
1             76              5.9375
2             66              4.7500
3            103              6.2500
4             76              5.5625
5            109              2.3750
6             98              5.1250

R> femaleArtistic = (feelings %>% filter(Gender=="Female") %>%
+  filter(Condition=="Abstract Art"))$AveragePleasantness
R> maleArtistic = (feelings %>% filter(Gender=="Male") %>%
+  filter(Condition=="Abstract Art"))$AveragePleasantness
R> femaleNude = (feelings %>% filter(Gender=="Female") %>%
+  filter(Condition=="Nudes"))$AveragePleasantness
R> maleNude = (feelings %>% filter(Gender=="Male") %>%
+  filter(Condition=="Nudes"))$AveragePleasantness
```

Second, the model assumption of normality in each group is checked:

```
R> assumption.check(femaleArtistic, maleArtistic, femaleNude, maleNude)
```

```
1: In assumption.check(femaleArtistic, maleArtistic, femaleNude, maleNude) :
  Model assumption of normally distributed data in each group is violated.
 All results of the Bayesian ANOVA based on a Gaussian mixture
 could therefore be unreliable and not trustworthy.
2: In assumption.check(femaleArtistic, maleArtistic, femaleNude, maleNude) :
  Run further diagnostics (like Quantile-Quantile-plots) to check if the
  Bayesian ANOVA can be expected to be robust to the violations of normality
```

This time the function gives a warning, that there are violations of the distributional assumptions. Investigating the results leads to the conclusion that data in the fourth group deviate from normality, shown in 5 in the QQ-plot. Still, as all other groups show no strong deviations from normality, we proceed and are cautious when drawing inferences including any statements involving the fourth group.

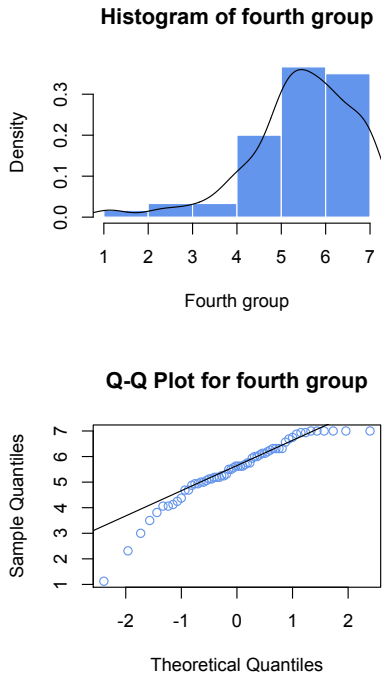


Figure 5: Histogram and quantile-quantile plot for the fourth group in the `feelings` dataset, showing that the assumption of normality is violated

Keeping this in mind, the Bayesian ANOVA is run now with default hyperparameters:

```
R> set.seed(42)
R> resFeelings = bayes.anova(first = femaleArtistic, second = maleArtistic,
+   third = femaleNude, fourth = maleNude)
```

Parameter	LQ	Mean	UQ	Std.Err
mu1	4.86	4.9	4.95	0.02
mu2	4.62	4.66	4.69	0.02
mu3	4.07	4.2	4.34	0.07
mu4	5.42	5.47	5.53	0.03
sigma1	0.98	1.16	1.4	0.11
sigma2	0.86	1.02	1.21	0.09
sigma3	1.34	1.66	2.06	0.19
sigma4	1.06	1.26	1.52	0.12
mu2-mu1	-0.31	-0.25	-0.19	0.03
mu3-mu1	-0.85	-0.7	-0.56	0.07
mu4-mu1	0.5	0.57	0.64	0.04
mu3-mu2	-0.6	-0.46	-0.32	0.07
mu4-mu2	0.75	0.81	0.87	0.03

```
|mu4-mu3       |1.12  |1.27  |1.41  |0.07    |
|sigma2-sigma1 |-0.43 |-0.14 |0.12  |0.14    |
|sigma3-sigma1 |0.1   |0.49  |0.95  |0.21    |
|sigma4-sigma1 |-0.21 |0.1   |0.42  |0.16    |
|sigma3-sigma2 |0.27  |0.64  |1.07  |0.21    |
|sigma4-sigma2 |-0.04 |0.24  |0.55  |0.15    |
|sigma4-sigma3 |-0.84 |-0.39 |0.01  |0.22    |
|delta12       |0.18  |0.24  |0.29  |0.03    |
|delta13       |0.47  |0.6   |0.73  |0.07    |
|delta14       |-0.58 |-0.52 |-0.44 |0.04    |
|delta23       |0.27  |0.41  |0.53  |0.06    |
|delta24       |-0.84 |-0.76 |-0.69 |0.04    |
|delta34       |-1.2  |-1.07 |-0.91 |0.07    |
```

The results show that differences are now much more subtle than in the previous examples. From the results one can spot that the means in the first three groups are located nearer to each other than in the previous examples, and the fourth group differs more strongly from the first three. The standard deviations do not differ a lot between groups, and the magnitude of the posterior effect sizes is now smaller, too. To investigate the effect sizes, visualisations are produced first:

```
R> anovaplot(resFeelings)
```

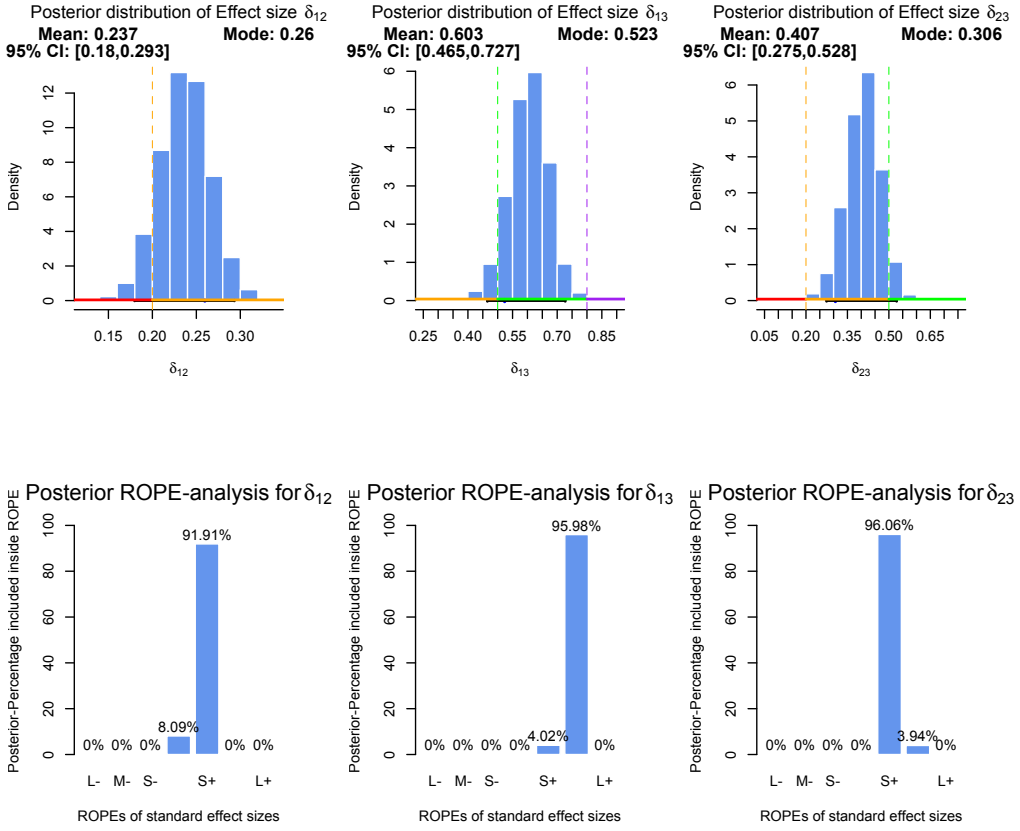


Figure 6: Visualisations of the posterior effect sizes for the `feelings` dataset using the `anovaplot()` function in **bayesanova**, showing which effects are most probable a posteriori based on a ROPE-analysis for each pair of groups

Figure 6 shows the plots produces by the above call to `anovaplot()`. The two left plots show that with 91.91% probability there is a small effect between the first and second group given the data, which are the female and male artistic pictures groups. Therefore, with large probability females rate artistic pictures more pleasant than males, where the effect size itself is small. Still, we could argue that there is nonnegligible probability of 8.09% that there is no effect at all and therefore not draw any conclusion depending on the posterior probability we require.

The middle two plots in 6 show the effect between the female artistic and female nude picture groups. We can see that based on the posterior distribution of δ_{13}, with 95.98% there is a medium effect between the two groups given the data. Females rate artistic pictures therefore with a probability near certainty as more pleasant than nude pictures, where the effect size in terms of standardized differences between ratings is medium.

The right two plots in 6 show the effect between the male artistic and female nude groups. The posterior reveals that 96.06% indicate a small effect, which could be interpreted as the fact that males rate artistic pictures even more pleasant than females rate nude pictures, but the effect size is only small and the remaining 3.94% posterior probability indicate that there is even a medium effect.

Figure 7 shows the effects which include the fourth group. Due to the violations of distributional assumptions one need to be cautious now, as the results could be deterred. Still, the two right plots show the effect size between the female and male nude groups, and indicate that the full posterior (100%) signals a large negative effect. This means, males rate the pleasantness of nude pictures much higher than females. Still, the result (as well as the results for δ_{14} and δ_{24}) are questionable due to the violation of model assumptions, so we do not proceed here.

The posterior predictive check in the right plot of 4 obtained via

```
post.pred.check(anovafit = resFeelings, ngroups = 4, out = feelings$AveragePleasantness,
reps = 100, eta = c(58/223,64/223,41/223,60/223))
```

shows that the overall fit seems reasonable, although there is some room for improvement in the range of average pleasantness ratings between zero and two, and in the peak between average pleasantness ratings of four and six. Subdividing the data even further and refitting the ANOVA model with a higher number of components would be an option to improve the fit. Alternatively, one could discuss the prior hyperparameters chosen here.

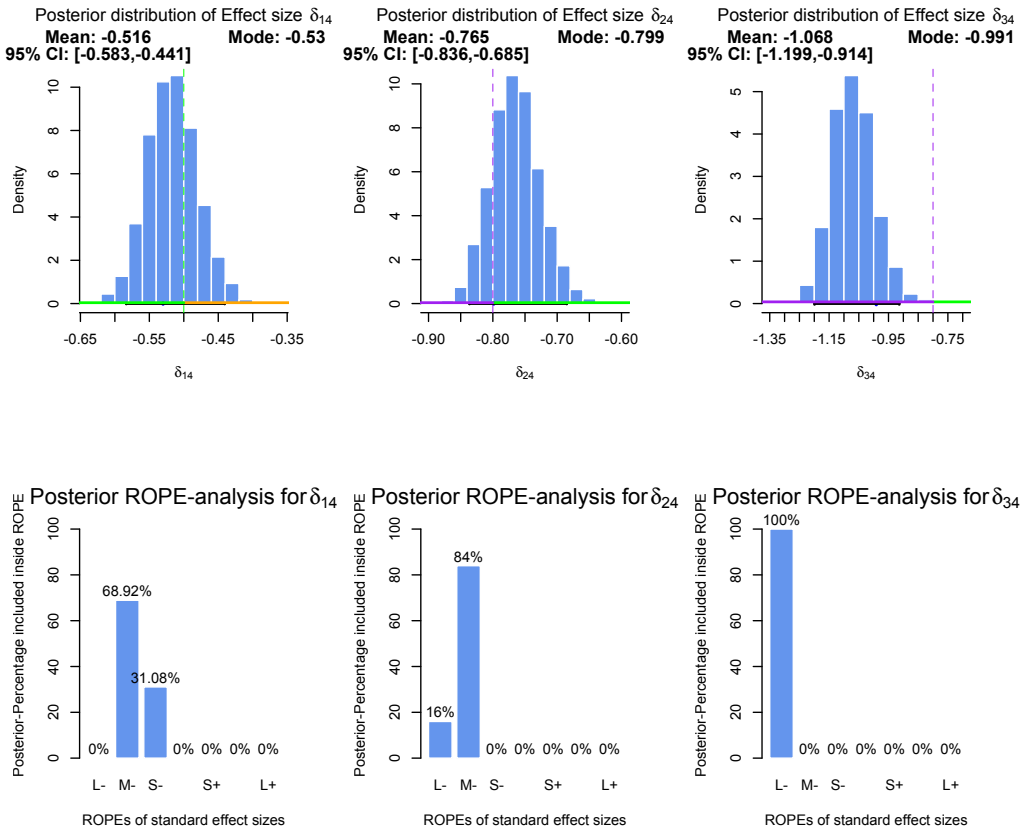


Figure 7: Visualisations of the posterior effect sizes for the `feelings` dataset using the `anovaplot()` function in **bayesanova**, showing which effects are most probable a posteriori based on a ROPE-analysis for each pair of groups

A solution via a traditional ANOVA in this case would yield:

```
R> summary(aov(AveragePleasantness ~ Gender * Condition, data = feelings))

                 Df Sum Sq Mean Sq F value   Pr(>F)
Gender            1  10.63  10.629   7.605  0.00631 **
Condition         1   1.27   1.267   0.906  0.34210
Gender:Condition   1  31.15  31.155  22.291 4.18e-06 ***
Residuals        219 306.09   1.398
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here, the condition is not significant, but the interaction is. The above analysis via the Bayesian mixture ANOVA made this more explicit: The posteriors for each combination of gender and condition were derived via MCMC, leading for example to the conclusion that females rate artistic picture as more pleasant than nude pictures with 95.98% probability for a medium effect size and 4.02% for a small effect size.

A solution via a Bayes factor based ANOVA would yield:

```
R> library(BayesFactor)
R> set.seed(42)
R> feelings$Gender = factor(feelings$Gender)
R> feelings$Condition = factor(feelings$Condition)
R> bfFeelings = anovaBF(AveragePleasantness ~ Gender * Condition, data = feelings)

Bayes factor analysis
--------------
[1] Gender                              : 3.727898   +- 0%
[2] Condition                           : 0.2532455  +- 0.01%
[3] Gender + Condition                  : 0.822604   +- 1.01%
[4] Gender + Condition + Gender:Condition : 3048.134   +- 1.14%

Against denominator:
  Intercept only
---
Bayes factor type: BFlinearModel, JZS
```

The conclusions drawn in this case are that the model including both gender, the condition and the interaction between both is most favourable due to the huge Bayes factor of $BF(\mathcal{M}_4, \mathcal{M}_0) = 3048.134$. Here too, the information is quite limited compared to the detailed analyses we could obtain from the Bayesian ANOVA based on the Gaussian mixture model above.

Amyloid concentrations and cognitive impairments

This example uses data from medical research about Alzheimer's disease. Amyloid-beta (Abeta) is a protein fragment which has been linked frequently to Alzheimer's disease. Autopsies from a sample of Catholic priests included measurements of Abeta (pmol/g tissue from the posterior cingulate cortex) from three groups: subjects who had exhibited no cognitive impairment before death, subjects who had exhibited mild cognitive impairment, and subjects who had mild to moderate Alzheimer's disease. The original study results were published by Pivtoraiko et al. (2015) in the journal *Neurobiology of Aging* and are used here.

The `Amyloid` dataset is available in the **Stat2Data** package (Cannon et al., 2019) and includes a group indicator `Group`, which takes either one of three values: `mAD`, which classifies a subject as having had mild Alzheimer's disease, `MCI`, which is a mild cognitive impairment and `NCI`, which is no cognitive impairment. Also, the amount of Amyloid-beta from the posterior cingulate cortex is given in pmol per gram tissue in the variable `Abeta`.

After loading and splitting the data into the three groups, we run the `assumption.check()` function:

```
R> library(Stat2Data)
R> data(Amyloid)
R> head(Amyloid)

  Group Abeta
1   NCI   114
2   NCI    41
3   NCI   276
```

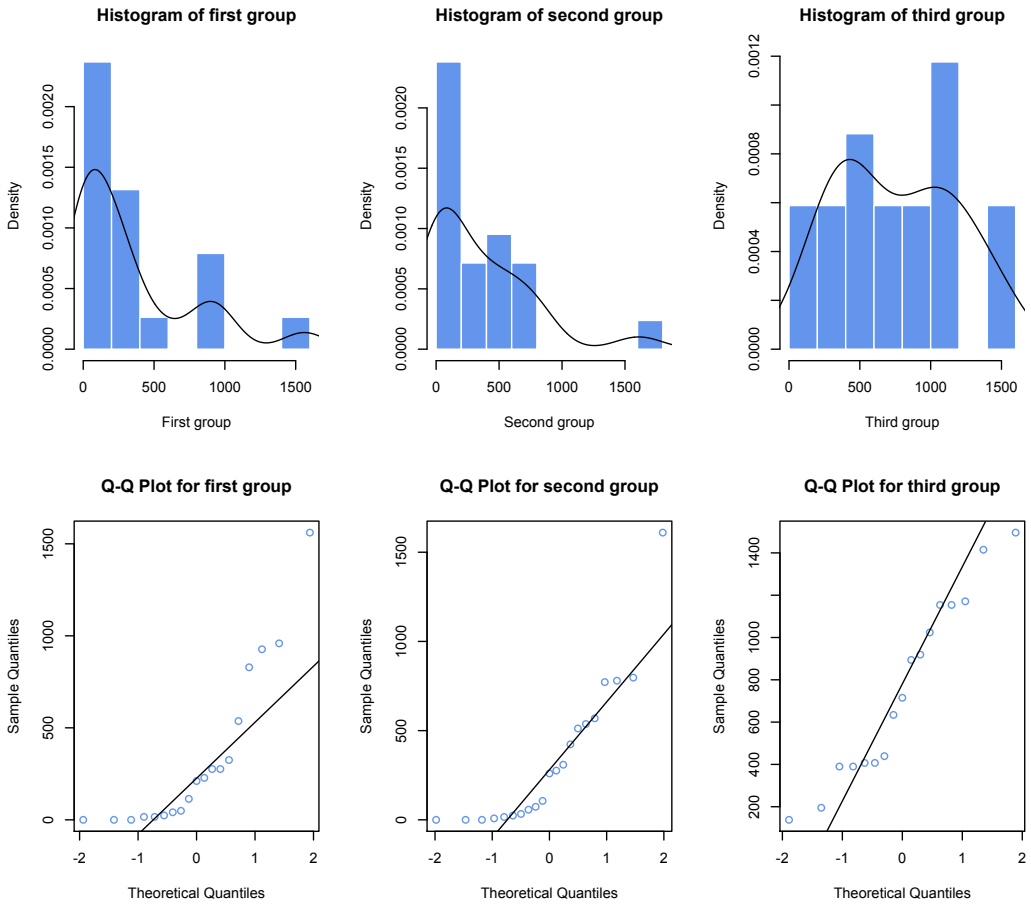


Figure 8: Model assumption checks for the `Amyloid` dataset using the `assumption.check()` function in **bayesanova**, showing that the assumption of a Gaussian mixture model is violated

```
4   NCI     0
5   NCI    16
6   NCI   228

R> NCI = (Amyloid %>% filter(Group=="NCI"))$Abeta
R> MCI = (Amyloid %>% filter(Group=="MCI"))$Abeta
R> mAD = (Amyloid %>% filter(Group=="mAD"))$Abeta
R> assumption.check(NCI, MCI, mAD)

1: In assumption.check(NCI, MCI, mAD) :
  Model assumption of normally distributed data in each group is violated.
 All results of the Bayesian ANOVA based on a multi-component Gaussian
 mixture could therefore be unreliable and not trustworthy.
2: In assumption.check(NCI, MCI, mAD) :
  Run further diagnostics (like Quantile-Quantile-plots) to check if the
  Bayesian ANOVA can be expected to be robust to the violations of normality
```

The results in 8 clearly show that the model assumptions are violated. Therefore, it is not recommended to run a Bayesian ANOVA in this case. A solution via a traditional ANOVA or via a Bayes factor based ANOVA would not proceed at this point, too.

A small simulation study – Recapturing simulation parameters of synthetic datasets

The next example is more in the veins of a simulation approach. We simulate three-, four-, five- and six-component Gaussian mixtures with increasing means $\mu_j := j$ and $\sigma_j = 1$. Therefore, the theoretical parameter values as well as the differences in means and standard deviations and the effect sizes δ_{lr} are known $\forall l, r$. We simulate 500 datasets with $n = 50$ observations in each group for each Gaussian

mixture above, and run the Bayesian ANOVA with default hyperparameters, that is 10000 Gibbs steps with 5000 burn-in steps, 95% credibility level and standard deviation output. Histograms of the posterior means for all parameters are shown in 9.

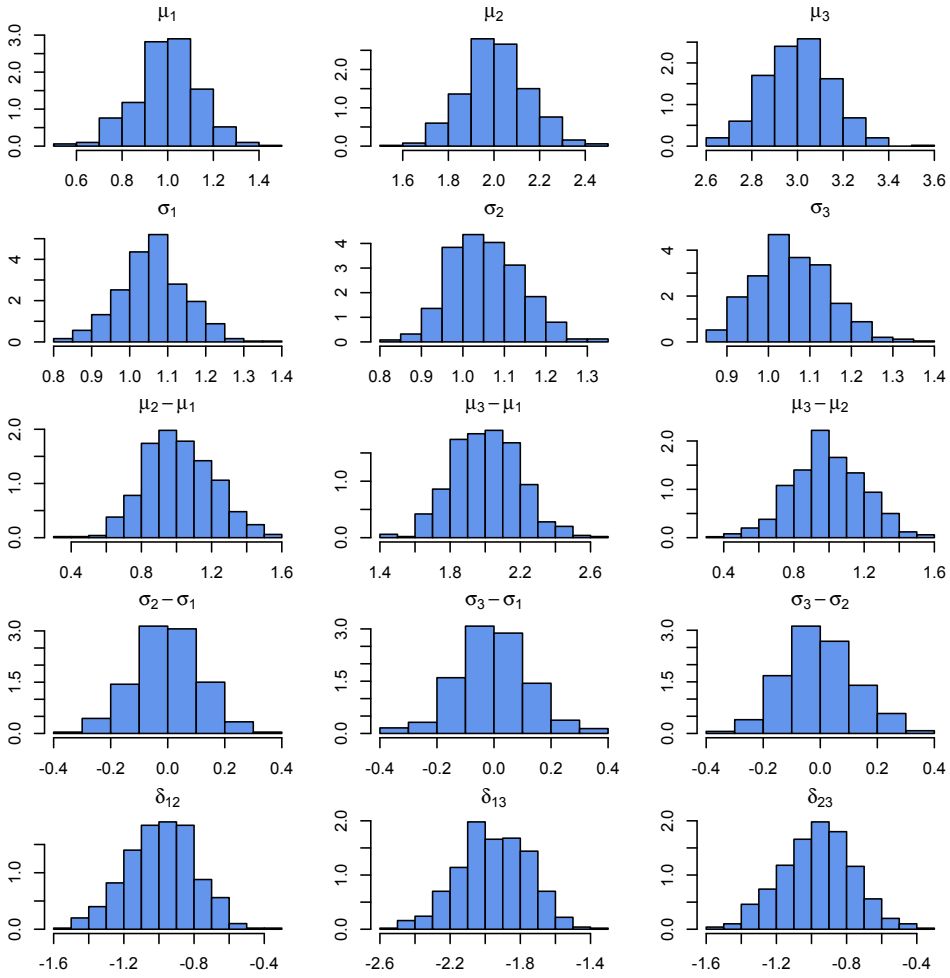


Figure 9: Recapturing simulation parameters of synthetic datasets with `bayes.anova()`, showing that the Gibbs sampler yields consistent estimates of the underlying effect sizes

The results clearly show that even for 500 simulated datasets the true parameters $\mu_j = j$ and $\sigma_j = 1$ are recaptured for small sample sizes like $n = 50$ in each group. Also, the differences in means $\mu_l - \mu_r, l \neq r$ are near one, and the differences in standard deviations $\sigma_l - \sigma_r, l \neq r$ are near zero. The effect sizes $\delta_{lr}, l \neq r$ also are recaptured as expected. More details about the theoretical properties of the procedure, especially the derivation of the Gibbs sampler for the two-group case can be found in Kelter (2020c, 2021d). Note that increasing sample sizes in the groups will yield consistent estimates as a result of MCMC theory Robert and Casella (2004).

7 Conclusion

This paper introduces **bayesanova**, an R package for conducting a Bayesian analysis of variance based on MCMC in a Gaussian mixture distribution with known allocations. The Bayesian ANOVA implemented in **bayesanova** is based on Gibbs sampling and supports up to six distinct components, which covers the typical range of ANOVAs used in empirical research.

The package provides four functions to check the model assumptions, run the Bayesian ANOVA, visualize the results and check the posterior fit. All functions have a variety of optional parameters to adapt them to a specific workflow or goal. Also, convergence issues can be detected via the built-in convergence diagnostics of all MCMC results in the `anovaplot()` function and it is possible to

post-process the results delivered as raw Markov chain draws by `bayes.anova`, for example via the R package **bayestestR** (Makowski et al., 2019a).

In the paper, multiple examples from medical and psychological research using real datasets were provided, showing the richness of information provided by the proposed procedure. Also, while explicit testing (for example via Bayes factors) is not implemented as standard output, it is worth noting that computing Bayes factors numerically based on the Gaussian mixture model is possible for example by using numerical techniques such as the Savage-Dickey density ratio (Kelter, 2021a; Wagenmakers et al., 2010; Dickey and Lientz, 1970; Verdinelli and Wasserman, 1995). However, the focus of explicit hypothesis testing is replaced in the default output of the procedure by estimation of the effect sizes between groups (or component density parameters) under uncertainty. If hypothesis testing is needed, the implemented ROPE can be used for rejecting a hypothesis based on interval hypothesis tests – compare Kelter (2021b), Linde et al. (2020) and Kruschke (2018) – or by using external packages like **bayestestR** (Makowski et al., 2019a) in conjunction with the raw samples provided by `bayes.anova`. Also, other indices like the probability of direction (Makowski et al., 2019b) or the MAP-based p-value (Mills, 2018) can be obtained via the package **bayestestR** (Makowski et al., 2019a) if hypothesis testing is desired, for an overview see Kelter (2021a). To offer users the freedom of choice for their preferred statistical evidence measure, only a ROPE-based estimate of the maximum a posteriori effect size δ is provided in **bayesanova**.

A small simulation study showed for the case of three-component Gaussian mixtures, that the provided MCMC algorithm precisely captures the true parameter values. Similar results hold for the four- or more-component case, as can easily be checked by adapting the provided R code.

In summary, the **bayesanova** package provides a novel and easy to apply alternative to existing packages like **stats** (R Core Team, 2020) or **BayesFactor** (Morey and Rouder, 2018), which implement the traditional frequentist ANOVA and Bayesian ANOVA models based on the Bayes factor.

Future plans include to add prior predictive checks and up to 12-component support, allowing for 2×6 Bayesian ANOVAs. Also, nonparametric mixtures could be applied in the case the model assumptions are violated, but therefore first theoretical results are necessary.

Bibliography

R. N. Balzarini, K. Bobson, K. Chin, and L. Campbell. Does exposure to erotica reduce attraction and love for romantic partners in men? Independent replications of Kenrick, Gutierres, and Goldberg (1989). *Journal of Experimental Social Psychology*, 70:191–197, 2017. [p67]

D. J. Benjamin, J. Berger, M. Johannesson, B. A. Nosek, E.-J. Wagenmakers, R. Berk, K. A. Bollen, B. Brembs, L. Brown, C. Camerer, D. Cesarini, C. D. Chambers, M. Clyde, T. D. Cook, P. De Boeck, Z. Dienes, A. Dreber, K. Easwaran, C. Efferson, E. Fehr, F. Fidler, A. P. Field, M. Forster, E. I. George, R. Gonzalez, S. Goodman, E. Green, D. P. Green, A. G. Greenwald, J. D. Hadfield, L. V. Hedges, L. Held, T. Hua Ho, H. Hoijtink, D. J. Hruschka, K. Imai, G. Imbens, J. P. A. Ioannidis, M. Jeon, J. H. Jones, M. Kirchler, D. Laibson, J. List, R. Little, A. Lupia, E. Machery, S. E. Maxwell, M. McCarthy, D. A. Moore, S. L. Morgan, M. Munafó, S. Nakagawa, B. Nyhan, T. H. Parker, L. Pericchi, M. Perugini, J. Rouder, J. Rousseau, V. Savalei, F. D. Schönbrodt, T. Sellke, B. Sinclair, D. Tingley, T. Van Zandt, S. Vazire, D. J. Watts, C. Winship, R. L. Wolpert, Y. Xie, C. Young, J. Zinman, and V. E. Johnson. Redefine statistical significance. *Nature Human Behaviour*, 2(1):6–10, 2018. ISSN 2397-3374. doi: 10.1038/s41562-017-0189-z. [p54]

J. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, New York, 1985. ISBN 9781441930743. [p56]

D. V. D. Bergh, J. V. Doorn, M. Marsman, K. N. Gupta, A. Sarafoglou, G. Jan, A. Stefan, A. Ly, and M. Hinne. A Tutorial on Conducting and Interpreting a Bayesian ANOVA in JASP. *psyarxiv preprint, https://psyarxiv.com/spreb*, 2019. [p54, 58]

A. Cannon, G. Cobb, B. Hartlaub, J. Legler, R. Lock, T. Moore, A. Rossman, and J. Witmer. Stat2Data: Datasets for Stat2, 2019. [p57, 71]

B. Carlin and T. Louis. *Bayesian Methods for Data Analysis*. Chapman & Hall, CRC Press, Boca Raton, sep 2009. [p54]

B. Carpenter, J. Guo, M. D. Hoffman, M. Brubaker, A. Gelman, D. Lee, B. Goodrich, P. Li, A. Riddell, and M. Betancourt. Stan : A Probabilistic Programming Language. *Journal of Statistical Software*, 76 (1):1–32, 2017. ISSN 1548-7660. doi: 10.18637/jss.v076.i01. [p55, 58]

J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Routledge, Hillsdale, N.J, 2nd edition, 1988. ISBN 978-0-8058-0283-2. [p56, 60, 62, 65]

D. Colquhoun. The reproducibility of research and the misinterpretation of p-values. *Royal Society Open Science*, 4(12), 2017. ISSN 20545703. doi: 10.1098/rsos.171085. [p54]

D. Colquhoun. The False Positive Risk: A Proposal Concerning What to Do About p-Values. *The American Statistician*, 73(sup1):192–201, 2019. ISSN 0003-1305. doi: 10.1080/00031305.2018.1529622. [p54]

J. M. Dickey and B. P. Lientz. The Weighted Likelihood Ratio, Sharp Hypotheses about Chances, the Order of a Markov Chain. *Annals of Mathematical Statistics*, 41(1):214–226, 1970. ISSN 0003-4851. doi: 10.1214/AOMS/1177697203. [p74]

T. S. Donaldson. Power of the F-test for nonnormal distributions and unequal error variances, 1966. [p55]

C. Dong and M. Wedel. BANOVA: Hierarchical Bayesian ANOVA Models, 2019. URL https://cran.r-project.org/package=BANOVA. [p58]

S. Frühwirth-Schnatter. *Finite mixture and Markov switching models*. Springer, New York, 2006. ISBN 9781441921949. [p58, 59]

J. Gabry, D. Simpson, A. Vehtari, M. Betancourt, and A. Gelman. Visualization in Bayesian workflow. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 182(2):389–402, feb 2019. ISSN 09641998. doi: 10.1111/rssa.12378. URL http://doi.wiley.com/10.1111/rssa.12378. [p62]

A. Gelman and S. P. Brooks. General Methods for Monitoring Convergence of Iterative Simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455, 1998. [p62]

A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, Cambridge, 2006. ISBN 9780511790942. doi: 10.1017/CBO9780511790942. URL http://ebooks.cambridge.org/ref/id/CBO9780511790942. [p57]

JASP Team. Jeffreys Awesome Statistics Package (JASP). *https://jasp-stats.org/*, 2019. URL https://jasp-stats.org/. [p54]

H. Jeffreys. *Theory of Probability*. Oxford University Press, Oxford, 3rd edition, 1961. ISBN 0-19-850368-7. [p56]

K. Kamary, K. Mengersen, C. P. Robert, and J. Rousseau. Testing hypotheses via a mixture estimation model. *arXiv preprint, https://arxiv.org/abs/1412.2044*, pages 1–37, 2014. ISSN 00237205. doi: 10.16373/j.cnki.ahr.150049. [p54, 56]

R. Kelter. Bayesian alternatives to null hypothesis significance testing in biomedical research: a non-technical introduction to Bayesian inference with JASP. *BMC Medical Research Methodology*, 20 (1), 2020a. ISSN 1471-2288. doi: 10.1186/s12874-020-00980-6. [p56]

R. Kelter. Analysis of Bayesian posterior significance and effect size indices for the two-sample t-test to support reproducible medical research. *BMC Medical Research Methodology*, 20(88), 2020b. doi: https://doi.org/10.1186/s12874-020-00968-2. [p56]

R. Kelter. bayest: An R Package for effect-size targeted Bayesian two-sample t-tests. *Journal of Open Research Software*, 8(14), 2020c. doi: https://doi.org/10.5334/jors.290. [p60, 61, 73]

R. Kelter. How to Choose between Different Bayesian Posterior Indices for Hypothesis Testing in Practice. *Multivariate Behavioral Research*, (in press):1–29, 2021a. ISSN 0027-3171. doi: 10.1080/00273171.2021.1967716. URL https://www.tandfonline.com/doi/full/10.1080/00273171.2021.1967716. [p74]

R. Kelter. Bayesian Hodges-Lehmann tests for statistical equivalence in the two-sample setting: Power analysis, type I error rates and equivalence boundary selection in biomedical research. *BMC Medical Research Methodology*, 21(1), 2021b. ISSN 1471-2288. doi: 10.1186/s12874-021-01341-7. [p74]

R. Kelter. On the Measure-Theoretic Premises of Bayes Factor and Full Bayesian Significance Tests: a Critical Reevaluation. *Computational Brain & Behavior*, (online first):1–11, 2021c. ISSN 2522-0861. doi: 10.1007/s42113-021-00110-5. [p56]

R. Kelter. A new Bayesian two-sample t-test and solution to the Behrens-Fisher problem based on Gaussian mixture distributions. *Statistics in Biosciences*, (in press), 2021d. [p60, 61, 73]

J. K. Kruschke. Bayesian estimation supersedes the t-test. *Journal of Experimental Psychology: General*, 142(2):573–603, 2013. ISSN 1939-2222. doi: 10.1037/a0029146. [p54]

J. K. Kruschke. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, Oxford, 2nd edition, 2015. ISBN 9780124058880. doi: 10.1016/B978-0-12-405888-0.09999-2. [p54, 55, 56, 57, 58, 60]

J. K. Kruschke. Rejecting or Accepting Parameter Values in Bayesian Estimation. *Advances in Methods and Practices in Psychological Science*, 1(2):270–280, 2018. doi: 10.1177/2515245918771304. [p54, 60, 74]

M. Linde, J. Tendeiro, R. Selker, E.-J. Wagenmakers, and D. van Ravenzwaaij. Decisions About Equivalence: A Comparison of TOST, HDI-ROPE, and the Bayes Factor. *psyarxiv preprint, https://psyarxiv.com/bh8vu*, 2020. [p74]

D. Makowski, M. Ben-Shachar, and D. Lüdecke. bayestestR: Describing Effects and their Uncertainty, Existence and Significance within the Bayesian Framework. *Journal of Open Source Software*, 4(40): 1541, 2019a. doi: 10.21105/joss.01541. [p74]

D. Makowski, M. S. Ben-Shachar, S. H. A. Chen, and D. Lüdecke. Indices of Effect Existence and Significance in the Bayesian Framework. *Frontiers in Psychology*, 10:2767, 2019b. ISSN 1664-1078. doi: 10.3389/fpsyg.2019.02767. [p54, 74]

R. McElreath and P. E. Smaldino. Replication, communication, and the population dynamics of scientific discovery. *PLoS ONE*, 10(8):1–16, 2015. ISSN 19326203. doi: 10.1371/journal.pone.0136088. [p54]

J. A. Mills. Objective Bayesian Precise Hypothesis Testing. Technical report, University of Cincinnati, 2018. [p74]

D. S. Moore, G. P. McCabe, and B. A. Craig. *Introduction to the practice of statistics*. W. H. Freeman, New York, 9th edition, 2012. ISBN 1319013384. [p64]

R. D. Morey and J. N. Rouder. BayesFactor: Computation of Bayes Factors for Common Designs. *R package version 0.9.12-4.2*, 2018. URL https://cran.r-project.org/package=BayesFactor. [p57, 74]

V. N. Pivtoraiko, E. E. Abrahamson, S. E. Leurgans, S. T. DeKosky, E. J. Mufson, and M. D. Ikonomovic. Cortical pyroglutamate amyloid-β levels and cognitive decline in Alzheimer's disease. *Neurobiology of Aging*, 36(1):12–19, jan 2015. ISSN 15581497. doi: 10.1016/j.neurobiolaging.2014.06.021. URL http://www.ncbi.nlm.nih.gov/pubmed/25048160http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4268150. [p71]

M. Plummer. JAGS: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, 2003. doi: 10.1111/j.2517-6161.1996.tb02070.x. [p55, 56, 57, 58]

R Core Team. R: A Language and Environment for Statistical Computing, 2020. URL https://www.r-project.org/. [p57, 74]

A. Raftery. Hypothesis testing and model selection. In W. Gilks, D. Spiegelhalter, and S. Richardson, editors, *Markov Chain Monte Carlo in Practice*, pages 163–188. Chapman & Hall, London, 1996. [p61]

C. R. Rao and M. M. Lovric. Testing point null hypothesis of a normal mean and the truth: 21st Century perspective. *Journal of Modern Applied Statistical Methods*, 15(2):2–21, 2016. ISSN 15389472. doi: 10.22237/jmasm/1478001660. [p56]

C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, New York, 2004. ISBN 1441919392. [p58, 61, 73]

C. P. Robert. The expected demise of the Bayes factor. *Journal of Mathematical Psychology*, 72(2009): 33–37, 2016. ISSN 10960880. doi: 10.1016/j.jmp.2015.08.002. [p54, 56]

J. Rochon, M. Gondan, and M. Kieser. To test or not to test: Preliminary assessment of normality when comparing two independent samples. *BMC Medical Research Methodology*, 12, 2012. ISSN 14712288. doi: 10.1186/1471-2288-12-81. [p54]

J. N. Rouder, R. D. Morey, P. L. Speckman, and J. M. Province. Default Bayes factors for ANOVA designs. *Journal of Mathematical Psychology*, 56(5):356–374, 2012. ISSN 00222496. doi: 10.1016/j.jmp.2012.08.001. URL http://dx.doi.org/10.1016/j.jmp.2012.08.001. [p54, 55, 56, 57, 60, 61]

S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965. URL http://www.jstor.org/stable/2333709. [p62]

Stan Development Team. RStan: the R interface to Stan. *R package version 2.19.3*, 2020. URL http://mc-stan.org/. [p55, 58]

J. N. Tendeiro and H. A. Kiers. A Review of Issues About Null Hypothesis Bayesian Testing. *Psychological Methods*, 24(6):774–795, 2019. ISSN 1082989X. doi: 10.1037/met0000221. [p56]

M. Tiku. Power Function of the F-test Under Non-Normal Situations. *Journal of the American Statistical Association*, 66(336), 1971. [p55]

J. van Doorn, D. van den Bergh, U. Bohm, F. Dablander, K. Derks, T. Draws, N. J. Evans, Q. F. Gronau, M. Hinne, S. Kucharský, A. Ly, M. Marsman, D. Matzke, A. Raj, A. Sarafoglou, A. Stefan, J. G. Voelkel, and E.-J. Wagenmakers. The JASP Guidelines for Conducting and Reporting a Bayesian Analysis. *psyarxiv preprint, https://psyarxiv.com/yqxfr*, 2019. doi: 10.31234/osf.io/yqxfr. [p54]

I. Verdinelli and L. Wasserman. Computing Bayes factors using a generalization of the Savage-Dickey density ratio. *Journal of the American Statistical Association*, 90(430):614–618, 1995. ISSN 1537274X. doi: 10.1080/01621459.1995.10476554. [p74]

E.-J. Wagenmakers, T. Lodewyckx, H. Kuriyal, and R. Grasman. Bayesian hypothesis testing for psychologists: A tutorial on the Savage-Dickey method. *Cognitive Psychology*, 60(3):158–189, 2010. ISSN 00100285. doi: 10.1016/j.cogpsych.2009.12.001. [p74]

R. L. Wasserstein and N. A. Lazar. The ASA's Statement on p-Values: Context, Process, and Purpose. *The American Statistician*, 70(2):129–133, 2016. ISSN 0003-1305. doi: 10.1080/00031305.2016.1154108. [p54]

R. L. Wasserstein, A. L. Schirm, and N. A. Lazar. Moving to a World Beyond "p<0.05". *The American Statistician*, 73(sup1):1–19, 2019. ISSN 0003-1305. doi: 10.1080/00031305.2019.1583913. [p54]

A. Zellner. Introduction. In A. Zellner and J. B. Kadane, editors, *Bayesian Analysis in Econometrics and Statistics : Essays in Honor of Harold Jeffreys*, chapter 1. Elsevier North-Holland, Amsterdam, 1980. ISBN 978-0444852700. [p56]

Riko Kelter
University of Siegen, Department of Mathematics
Walter-Flex-Street 3
57072, Siegen
Germany
ORCiD: 0000-0001-9068-5696
riko.kelter@uni-siegen.de

Appendix

Details on the F-statistic in frequentist ANOVA

After observing the data, the following quantities are calculated: For group j, $j = 1, ..., k$, I_j experimental units are observed and the empirical mean $m_j = 1/I_j \sum_{l=1}^{I_j} y_{lj}$ and empirical variance $s_j^2 = 1/(I_j - 1) \sum_{l=1}^{I_j} (y_{lj} - m_j)^2$ are calculated (data is assumed to be listed in a table where the groups correspond to the columns). The sum $\Sigma_{i \in I_j} y_{ij}$ and the sum of squares $\Sigma_{i \in I_j} (y_{ij})^2$ are calculated, to partition the variance into treatment and error sum of squares

$$SS_{Treatment} := \sum_{j=1}^{k} I_j (m_j - m)^2 \qquad SS_{Error} := \sum_{j=1}^{k} (I_j - 1) s_j^2 \tag{16}$$

$$SS_{Total} := \sum_{j=1}^{k} \sum_{i=1}^{n} (y_{ij} - m)^2 \tag{17}$$

where $m := 1/k \sum_{j=1}^{k} m_j$. Standard calculus yields that these sums of squares can be calculated as:

$$SS_{Treatment} := \sum_{j=1}^{k} \frac{(\sum_i y_{ij})^2}{I_j} - \frac{(\sum_j \sum_i y_{ij})^2}{I} \qquad SS_{Error} := \sum_{j=1}^{k} \sum_i y_{ij}^2 - \sum_j \frac{(\sum_i y_{ij})^2}{I_j} \tag{18}$$

$$SS_{Total} := \sum_{j=1}^{k} \sum_i y_{ij}^2 - \frac{(\sum_j \sum_i y_{ij})^2}{I} \tag{19}$$

Using the corresponding degrees of freedom $DF_{Treatment} = k - 1$, $DF_{Error} = n - k$ and $DF_{Total} = n - 1$, the F-statistic is defined as

$$F = \frac{MS_{Treatment}}{MS_{Error}} \tag{20}$$

where

$$MS_{Treatment} := \frac{SS_{Treatment}}{DF_{Treatment}} \qquad MS_{Error} := \frac{SS_{Error}}{DF_{Error}} \tag{21}$$

using only the quantities defined above. If the F-statistic is larger than the α-quantile for significance level α, H_0 is rejected.

tvReg: Time-varying Coefficients in Multi-Equation Regression in R

by Isabel Casas and Rubén Fernández-Casal

Abstract This article explains the usage of R package **tvReg**, publicly available for download from the Comprehensive R Archive Network, via its application to economic and finance problems. The six basic functions in this package cover the kernel estimation of semiparametric panel data, seemingly unrelated equations, vector autoregressive, impulse response, and linear regression models whose coefficients may vary with time or any random variable. Moreover, this package provides methods for the graphical display of results, forecast, prediction, extraction of the residuals and fitted values, bandwidth selection and nonparametric estimation of the time-varying variance-covariance matrix of the error term. Applications to risk management, portfolio management, asset management and monetary policy are used as examples of these functions usage.

1 Introduction

A very popular research area has been brewing in the field of kernel smoothing statistics applied to linear models with time-varying coefficients. In econometrics, Robinson (1989) was the first to analyse these models for linear regressions with time-varying coefficients and stationary variables. Since then, this literature has extended to models with fewer restrictions in the dependence of the variables to models with time dependence in the error term and to multi-equation models. Although these models are potentially applicable to a large number of areas, no comprehensive computational implementation is, to our knowledge, formally available in any of the commercial programming languages. The package **tvReg** contains the aforementioned functionality, input and output interface, and user-friendly documentation.

Parametric multi-equation linear models have increased in popularity in the last decades due to an increase in access to multiple datasets. Their application extends to, perhaps, every field of quantitative research. Just to mention some, they are found in biostatistics, finance, economics, business, climate, linguistics, psychology, engineering and oceanography. Panel linear models (PLM) are widely used to account for the heterogeneity in the cross-section and time dimensions. Seemingly unrelated equations (SURE) and vector autoregressive models (VAR) are the extensions of linear regressions and autoregressive models to the multi-equation framework. Programs with these algorithms are found in all major programming languages. Particularly in R, the package **plm** (Croissant and Millo, 2018, 2008) contains a comprehensive functionality for panel data models. The package **systemfit** (Henningsen and Hamann, 2007) allows the estimation of coefficients in systems of linear regressions, both with equation error terms correlated among equations (SURE) or uncorrelated. Finally, the package **vars** (Pfaff, 2008) provides the tools to fit VAR models and impulse response functions (IRF). All these functions assume that the coefficients are constant. This assumption might not be true when a time series runs for a long period, and the relationships among variables do change. The package **tvReg** is relevant in this case.

In comparison to parametric models, the appeal of nonparametric models is their flexibility and robustness to functional form misspecification, with spline-based and kernel-based regression methods being the two main nonparametric estimation techniques, (e.g. Eubank, 1999). However, fully nonparametric models are not appropriate when many regressors are in play, as their rate of convergence decreases with the number of regressors, the infamous "curse of dimensionality". In the case of cross-section data, a popular alternative to avoid this problem are the generalised additive models (GAM), introduced by Hastie and Tibshirani (1993). The GAM is a family of semiparametric models that extends parametric linear models by allowing for non-linear relationships of the explanatory variables and still retaining the additive structure of the model. In the case of time-series data, the most suitable alternative to nonparametric models is the linear models whose coefficients change over time or follow the dynamics of another random variable. This functionality is coded in R, within the single-equation framework, in packages **mgm** (Haslbeck and Waldorp, 2020), and **MARSS** (Holmes et al., 2012). Package **tvReg** uses the identical kernel smoothing estimation as package **mgm** when using a Gaussian kernel to estimate a VAR model with varying coefficients (TVVAR). However, the interpretation of their results is different because they are aimed at different audiences. The **mgm** focuses in the field of network models, producing network plots to represent relationships between current variables and their lags. Whereas the **tvReg** focuses in the field of economics where a direct interpretation of the TVVAR coefficients is not meaningful and may be done via the time-varying impulse response function (TVIRF) instead. Models with coefficients varying over time can also be

expressed in state space form, which assumes that the coefficients change over time in a determined way for example, as a Brownian motion. These models can be estimated using the Kalman filter or Bayesian techniques, for instance (Liu and Guo, 2020; Primiceri, 2005). Packages **MARSS** and **bvarsv** (Krueger, 2015) implement this approach based on the Carter and Kohn (1994) algorithm to estimate the TVVAR. On top of all this and as far as we can tell, the **tvReg** is the only package containing tools to estimate time-varying coefficients seemingly unrelated equation (TVSURE) and panel linear models (TVPLM) in R.

Simply, the main objective of the **tvReg** is to provide tools to estimate and forecast linear models with time-varying coefficients in the framework of kernel smoothing estimation, which may be difficult for the nonspecialised end-user to code. For completion, the **tvReg** also implements methods for the time-varying coefficients linear model (TVLM) and the time-varying coefficients autoregressive (TVAR) model. Often, these can be estimated using packages **gam** (Hastie, 2022) and **mgcv** (Wood, 2017), which combine (restricted) marginal likelihood techniques in combination with nonparametric methodologies. However, the advantage of using the **tvReg** is that it can handle dependency and any kind of distribution in the error term because it combines least squares techniques with nonparametric methodologies. An example of this is shown in Section Standard usage of `tvLM`.

Summing up, this paper presents a review of the most common time-varying coefficient linear models studied in the econometrics literature during the last two decades, their estimation using kernel smoothing techniques, the usage of functions and methods in the package **tvReg**, and their latest applications. Along these lines, Table 1 offers a glimpse at the **tvReg** full functionality, displaying a summary of its methods, classes and functions.

Function	Class	Function and Methods for class	Based on
`tvPLM`	`"tvPLM"`	`tvRE, tvFE, coef, confint, fitted, forecast, plot, predict, print, resid, summary`	`plm::plm`
`tvSURE`	`"tvsure"`	`tvGLS, bw, coef, confint, fitted, forecast, plot, predict, print, resid, summary`	`systemfit::systemfit`
`tvVAR`	`"tvvar"`	`tvAcoef, tvBcoef, tvIRF, tvOLS, tvPhi, tvPsi, bw, coef, confint, fitted, forecast, plot, predict, print, resid, summary`	`vars::VAR`
`tvIRF`	`"tvirf"`	`coef, confint, plot, print, summary`	`vars::irf`
`tvLM`	`"tvlm"`	`tvOLS, bw, coef, confint, fitted, forecast, plot, predict, print, resid, summary`	`stats::lm`
`tvAR`	`"tvar"`	`tvOLS, bw, coef, confint, fitted, forecast, plot, predict, print, resid, summary`	`stats::ar.ols`

Table 1: Structure of the package **tvReg**.

2 Multi-equation linear models with time-varying coefficients

A multi-equation model formed by a set of linear models is defined when each equation has its own dependent variable and possible different regressors. Seemingly unrelated equations, panel data models and vector autoregressive models are included in this category.

Time-varying coefficients SURE

The SURE was proposed by Zellner (1962) and is referred to as the seemingly unrelated equations model (SURE). The SURE model is useful to exploit the correlation structure between the error terms of each equation. Suppose that there are N linear regressions of different dependent variables,

$$Y_t = X_t\beta(z_t) + U_t \quad i = 1,\dots,N \quad t = 1,\dots,T, \tag{1}$$

where $Y_t = (y_{1t}\dots y_{Nt})^\top$ with $y_i = (y_{i1},\dots,y_{iT})^\top$ denotes the values over the recorded time period of the $i-$ th dependent variable. Each equation in (1) may have a different number of exogenous

variables, p_i. The regressors matrix, $X_t = diag(x_{1t} \dots x_{Nt})$ with $X_i = (x_{i1}, \dots, x_{ip_i})$ for equation i and $\beta_{z_t} = (\beta_1(z_t)^\top, ..., \beta_N(z_t)^\top)^\top$ is a vector of order $P = p_1 + p_2 + \dots + p_N$. The error vector, $U_t = (u_{1t} \dots u_{Nt})^\top$, has zero mean and covariance matrix $\mathbb{E}(U_t U_t^\top) = \Sigma_t$ with elements $\sigma_{ii't}$.

It is important to differentiate between two types of smoothing variables: 1) $z_t = \tau = t/T$ is the rescaled time with $\tau \in [0, 1]$, and 2) z_t is the value at time t of the random variable $Z = \{z_t\}_{t=1}^T$. In other words, time-varying coefficients may be defined as unknown functions of time, $\beta(z_t) = f(\tau)$, or as unknown functions of a random variable, $\beta(z_t) = f(z_t)$. The estimation of the TVSURE has been studied by Henderson et al. (2015) when for a random z_t and by Orbe et al. (2005) and Casas et al. (2019) for $z_t = \tau$. These estimators are consistent and asymptotically normal under certain assumptions on the size of the bandwidth, kernel regularity and error moments, and dependency. Details are left out of this text as can be easily found in the related literature.

The estimation of system (1) may be done separately for each equation as if there is no correlation in the error term across equations, i.e. system (1) has a total of N different TVLM with possibly N different bandwidths, b_i. In this case, the time-varying coefficients are obtained by combining the ordinary least squares (OLS) and the local polynomial kernel estimator, which is extensively studied in Fan and Gijbels (1996). The result is the time-varying OLS denoted by TVOLS herein. Two versions of this estimator are implemented in **tvReg**: i) the TVOLS that uses the local constant (*lc*) kernel method, also known as the Nadaraya-Watson estimator; and ii) the TVOLS which uses the local linear (*ll*) method. Focussing in the single equation i, and assuming that $\beta_i(\cdot)$ is twice differentiable, an approximation of $\beta_i(z_t)$ around z is given by the Taylor rule, $\beta_i(z_t) \approx \beta_i(z) + \beta_i^{(1)}(z)(z_t - z)$, where $\beta_i^{(1)}(z) = d\beta_i(z)/dz$ is its first derivative. The estimates resolve the following minimisation:

$$(\hat{\beta}_i(z_t), \hat{\beta}_i^{(1)}(z_t)) = \arg\min_{\theta_0, \theta_1} \sum_{t=1}^{T} \left[y_i - X_i^\top \theta_0 - (z_t - z) X_i^\top \theta_1 \right]^2 K_{b_i}(z_t - z).$$

Roughly, these methodologies fit a set of weighted local regressions with an optimally chosen window size. The size of these windows is given by the bandwidth b_i, and the weights are given by $K_{b_i}(z_t - z) = b_i^{-1} K(\frac{z_t - z}{b_i})$, for a kernel function $K(\cdot)$. The local linear estimator general expression is

$$\begin{pmatrix} \hat{\beta}_i(z_t) \\ \hat{\beta}_i^{(1)}(z_t) \end{pmatrix} = \begin{pmatrix} S_{T,0}(z_t) & S_{T,1}^\top(z_t) \\ S_{T,1}(z_t) & S_{T,2}(z_t) \end{pmatrix}^{-1} \begin{pmatrix} T_{T,0}(z_t) \\ T_{T,1}(z_t) \end{pmatrix} \tag{2}$$

with

$$\begin{aligned} S_{T,s}(z_t) &= \frac{1}{T} \sum_{i=1}^{T} X_i^\top X_i (z_i - z_t)^s K\left(\frac{z_i - z_t}{b_i}\right) \\ T_{T,s}(z_t) &= \frac{1}{T} \sum_{i=1}^{T} X_i^\top (z_i - z_t)^s K\left(\frac{z_i - z_t}{b_i}\right) y_i \end{aligned}$$

and $s = 0, 1, 2$. The particular case of the local constant estimator is calculated by $\hat{\beta}_{i,t} = S_{T,0}^{-1}(z_t) T_{T,0}(z_t)$ and it is only necessary that $\beta_i(\cdot)$ has one derivative.

A second option is to use the correlation matrix of the error term in the estimation of system (1). This is called the time-varying generalised least squares (TVGLS) estimation. Its mathematical expression is the same as (2) with the following matrix components:

$$\begin{aligned} S_{T,s}(z_t) &= \frac{1}{T} \sum_{i=1}^{T} X_i^\top K_{B,it}^{1/2} \Sigma_i^{-1} K_{B,it}^{1/2} X_i (Z_i - z_t)^s \\ T_{T,s}(z_t) &= \frac{1}{T} \sum_{i=1}^{T} X_i^\top K_{B,it}^{1/2} \Sigma_i^{-1} K_{B,it}^{1/2} Y_i (Z_i - z_t)^s, \end{aligned} \tag{3}$$

where $K_{B,it} = diag(K_{b_1,it}, ..., K_{b_N,it})$ and $K_{b_i,it} = (Tb_i)^{-1} K((Z_i - z_t)/(Tb_i))$ is the matrix of weights introducing smoothness according to the vector of bandwidths, $B = (b_1, \dots, b_N)^\top$. Note that this minimisation problem accounts for the time-varying structure of the variance-covariance matrix of the errors, Σ_t.

The TVGLS assumes that the error variance-covariance matrix is known. In practice, this is unlikely and it must be estimated, resulting in the Feasible TVGLS estimator (TVFGLS). This estimator consists of two steps:

Step 1 Estimate Σ_t based on the residuals of a line by line estimation (i.e, when Σ_t is the identity matrix). If Σ_t is known to be constant, the sample variance-covariance matrix from the residuals

is a consistent estimator of it. If Σ_t changes over time, a nonparametric estimator such the one explained in Section Estimating a time-varying variance-covariance matrix is a consistent alternative.

Step 2 Estimate the coefficients of the TVSURE by plugging in $\hat{\Sigma}_t$ from step 1 into Equation (3).

To ensure a good estimation of Σ_t, the iterative TVFLGS may be used. First, do steps 1-2 as above to obtain the residuals from step 2, and repeat step 2 until the estimates of Σ_t converge or the maximum number of iterations is reached.

Time-varying coefficients panel data models

Panel data linear models (PLM) are a particular case of SURE models with the same variables for each equation but measured for different cross-section units, such as countries, and for different points in time. All equations have the same coefficients apart from the intercept which can be different for different cross-sections. Therefore, the data from all cross-sections can be pooled together. The individual effects, α_i, account for the heterogeneity embedded in the cross-section dimension. This package only take into account balanced panel datasets, i.e. with the same number of data points for each cross-section unit.

Coefficient dynamics can be added to classical PLM models using a time-varying coefficients panel data model, TVPLM. Recent developments in this kind of models can be found in Sun et al. (2009); Dong, C. Jiti Gao, J. and Peng, B. (2015); Casas et al. (2021); Dong et al. (2021) among others, with general model,

$$y_{it} = \alpha_i + x_{it}^{\top}\beta(z_t) + u_{it} \quad i = 1, \ldots, N, \quad t = 1, \ldots, T. \tag{4}$$

Note that the smoothing variable only changes in the time dimension, not like in the SURE model where it changed over i and t. The three estimators of Equation (4) in the **tvReg** are:

1. The time-varying pooled ordinary least squares (TVPOLS) has the same expression than estimator (2) with the following terms:

$$\begin{aligned} S_{T,s}(z_t) =& X^{\top} K_{b,t}^{*} X (Z - z_t)^s \\ T_{T,s}(z_t) =& X^{\top} K_{b,t}^{*} Y (Z - z_t)^s, \end{aligned} \tag{5}$$

   where $K_{b,t}^{*} = I_N \otimes \text{diag}\{K_b(z_1 - z_t), \ldots, K_b(z_T - z_t)\}$. Note that it is not possible to ignore the panel structure in the semiparametric model because the coefficients change over time. The consistency and asymptotic normality of this estimator needs the classical assumptions about the kernel and the regularity of the coefficients, available in the related literature.

2. The time-varying random effects (TVRE) estimator is also given by Equation (5) with a non-identity Σ:

$$\begin{aligned} S_{T,s}(z_t) =& X^{\top} K_{b,t}^{*1/2} \Sigma_t^{-1} K_{b,t}^{*1/2} X (Z - z_t)^s \\ T_{T,s}(z_t) =& K_{b,t}^{*1/2} \Sigma_t^{-1} K_{b,t}^{*1/2} Y (Z - z_t)^s. \end{aligned} \tag{6}$$

   Note that this is a simpler case of (3) with the same bandwidth for all equations. The variance-covariance matrix is estimated in the same way using the residuals from the TVPOLS and it may be an iterative algorithm until convergence of the coefficients.

3. The time-varying fixed effects (TVFE) estimator. Unfortunately, the transformation for the within estimation does not work in the time-varying coefficients model because the coefficients depend on time (Sun et al., 2009, explain the issue in detail). Therefore, it is necessary to make the assumption that $\sum_{i=1}^{N} \alpha_i = 0$ for identification. The terms in the TVFE estimator are:

$$\begin{aligned} S_{T,s}(z_t) =& X^{\top} W_{b,t} X (Z - z_t)^s \\ T_{T,s}(z_t) =& X^{\top} W_{b,t} Y (Z - z_t)^s, \end{aligned} \tag{7}$$

   where $W_{b,t} = D_t^{\top} K_{b,t}^{*} D_t$, $D_t = I_{NT} - D(D^{\top} K_{b,t}^{*} D)^{-1} D^{\top} K_{b,t}^{*}$, $D = (-1_{N-1}, I_{N-1})^{\top} \otimes 1_T$, and 1_k is the unity vector of length k. The fixed effects are given by, $\hat{\alpha} = (D^{\top} K_{b,t}^{*} D)^{-1} D^{\top} K_{b,t}^{*} (Y - X^{\top}\beta)$. Finally, $\hat{\alpha}_i = \frac{1}{T} \sum_{t=1}^{T} \alpha_{it}$ for $i = 2, \ldots, N$.

Time-varying coefficient VAR model

Macroeconomic econometrics experienced a revolution when Sims (1980) presented the vector autoregressive (VAR) model: a new way of summarising relationships among several variables while

getting around the problem of endogeneity of structural models. The VAR model has lagged values of the dependent variable, y_t, as regressors to which further exogenous variables can be added as regressors. Unless the model is constrained, all variables are the same for every equation, which simplifies the algebra. The model coefficients and variance-covariance matrix may be estimated by maximum likelihood, OLS or GLS. VAR coefficients and the variance-covariance matrix do not have a direct economic interpretation. However, it is possible to use them to recover a structural model by imposing a number of restrictions and so analyse the transmission of a shock, for example, a new monetary policy, to the macroeconomy using the impulse response function (IRF). Lütkepohl (2005) dive into the theoretical properties of these models in detail.

The TVVAR(p) is an N-dimensional system of time-varying autoregressive processes of order p like

$$Y_t = A_{0,t} + A_{1,t}Y_{t-1} + \ldots + A_{p,t}Y_{t-p} + U_t, \;\; t = 1, 2, \ldots, T. \tag{8}$$

In Equation (8), $Y_t = (y_{1t}, \ldots, y_{Nt})^\top$ and coefficient matrices at each point in time $A_{j,t} = (a^j_{1t}, \ldots, a^j_{Nt})$, $j = 1, \ldots, p$ are of dimension $N \times N$. Then, notation $A_{j,t}$ means that the elements of this matrix are unknown functions of either the rescaled time value, τ, or of a random variable at time t. The innovation, $U_t = (u_{1t}, \ldots, u_{Nt})$, is an N-dimensional identically distributed random variable with $E(U_t) = 0$ and possibly a time-varying positive definite variance-covariance matrix, $E(U_t U_s^\top) = \Sigma_t$, for $t = s$, $E(U_t U_s^\top) = 0$ otherwise. Here, matrix $A_{j,t}$ is a function of τ, then process (8) is locally stationary in the sense of Dahlhaus (1997), which occurs when the functions in matrices $A_{j,t}$ are constant or change smoothly over time. Then, process (8) at time t has a well defined unique solution given by the Wold representation,

$$\bar{y}_t = \sum_{j=0}^{\infty} \Phi_{j,t} U_{t-j}, \tag{9}$$

such that $|Y_t - \bar{y}_t| \to 0$ almost surely. Matrix $\Phi_{0,t} = I_N$ and matrix $\Phi_{s,t} = \sum_{j=1}^{s} \Phi_{s-j,t} A_{j,t}$ for horizons $s = 1, 2, \ldots$ As for the constant model, $\Phi_{s,t}$ are the time-varying coefficient matrices of the impulse response function (TVIRF). Its element (t, i, j) may be interpreted as the expected response of $y_{i,t+s}$ to an exogenous shock of $y_{j,t}$ ceteris paribus lags of y_t when the innovations are orthogonal. Otherwise, an orthogonal TVIRF can be found as $\Psi_{j,t} = \Phi_{j,t} P_t$ for $\Sigma_t = P_t P_t^\top$, the Cholesky decomposition of Σ_t at time t. More theoretical details in Yan et al. (2021).

In the macroeconomic literature, the Bayesian estimation of process (8) has attracted a lot of attention in recent years driven by results in Cogley and Sargent (2005); Primiceri (2005) and Kapetanios et al. (2012). In their approach, the coefficients are assumed to follow a random walk. Recently, Kapetanios et al. (2017) studied the inference of the local constant estimator of a TVVAR(p) for large sets, and they found an increase in the forecast accuracy in comparison to the forecast accuracy of the VAR(p).

3 Standard usage of `tvSURE`

The main argument of this function is a list of formulas, one for each equation. The `formula` follows the format of `formula` in the package **systemfit**, which implements estimators of parametric multi-equation models with constant coefficients. The `tvSURE` wraps the `tvOLS` and `tvGLS` methods to estimate the coefficients of system (1). The `tvOLS` method is used by default, calculating estimates for each equation independently with different bandwidths, `bw`. The user is able to enter a set of bandwidths or a single bandwidth to be used in the estimation instead. The `tvGLS` method has argument `Sigma` where a known variance-covariance matrix of the error can be entered in Equation (3). Otherwise, if `Sigma = NULL`, the variance-covariance matrix Σ_t is estimated using function `tvCov`, which is discussed in Section Estimating a time-varying variance-covariance matrix.

In addition to `formula`, function `tvSURE` has other arguments to control and choose the desired estimation procedure:

Smoothing random variable
: All methods assume by default that the coefficients are unknown functions of $\tau = t/T$ and therefore argument `z` is set to `NULL`. The user can modify this setting by entering a numeric vector in argument `z` with the values of the random smoothing variable over the corresponding time period. Note that the current version only allows one single smoothing random variable, `z`, common for all equations; and balanced panels.

Bandwidth
: When argument `bw` is set to `NULL`, it is automatically selected by leave-one-out cross-validation. It is possible to select it by leave-k-out cross-validation (Chu and Marron, 1991) by setting argument `cv.block = k` (`k=0` by default). This minimisation can be slow for large datasets, and

it should be avoided if the user knows an appropriate value of the bandwidth for the required problem.

Kernel type
The three choices for this argument are `tkernel = "Triweight"` (default), `tkernel = "Epa"` and `tkernel = "Gaussian"`. The first two options refer to the Triweight and Epanechnikov kernels, which are compact in [-1, 1]. The authors recommend the use of either of those two instead of the Gaussian kernel which, in general, requires more calculations.

Degree of local polynomial
The default estimation methodology is the Nadaraya-Watson or local constant, which is set as (`est = "lc"`) and it fits a constant at each interval defined by the bandwidth. The argument `est = "ll"` can be chosen to perform a local linear estimation (i.e., to fit a polynomial of order 1).

Singular fit
The `tvOLS` method used in the estimation wraps the `lm.wfit` method, which at default allows the fitting of a low-rank model, and the estimation coefficients can be *NA*s. The user can change the argument `singular.ok` to `FALSE`, so that the program stops in case of a low-rank model.

The user can restrict certain coefficients in the TVSURE model using arguments `R` and `r`. Note that the restriction is done by setting those coefficients to a constant. Furthermore, argument `method` defines the type of estimator to be used. The possible choices in argument `method` are:

1. `"tvOLS"` for a line by line estimation, i.e, with Σ the identity matrix.
2. `"tvGLS"` to estimate the coefficients of the system using Σ_t, for which the user must enter it in argument `Sigma`. Argument `Sigma` takes either a symmetric matrix or an array. If `Sigma` is a matrix (constant over time) then it must have dimensions $neq \times neq$, where neq is the number of equations in the system. If Σ_t changes with time, then argument `Sigma` is an array of dimension $neq \times neq \times obs$, where the last dimension measures the number of time observations. Note that if the user enters a diagonal variance-covariance matrix with diagonal values different from one, then a time-varying weighted least squares is performed. If `method ="tvGLS"` is entered but `Sigma = NULL`, then `tvSURE` is fitted as if `method = "tvOLS"` and a warning is issued.
3. `"tvFGLS"` to estimate the coefficients of the system using an estimate of Σ_t. By default, only one iteration is performed in the estimation, unless argument `control` indicates otherwise. The user can choose the maximum number of iterations or the level of tolerance in the estimation of Σ_t. See example the below for details.

The package **systemfit** contains the `Kmenta` dataset, which was first described in Kmenta (1986), to show the usage of the function `systemfit` to fit SURE models. This example has two equations: i) a demand equation, which explains how food consumption per capita, `consump`, depends on the ratio of food price, `price`; and disposable income, `income`; and ii) a supply equation, which shows how consumption depends on `price`, ratio prices received by farmers to general consumer prices, `farmPrice`; and a possible time trend, `trend`. Mathematically, this SURE model is

$$\begin{aligned} consump_t &= \beta_{10} + \beta_{11} price_t + \beta_{12} income_t + u_{1t} \\ consump_t &= \beta_{20} + \beta_{21} price_t + \beta_{22} farmPrice_t + \beta_{23} t + u_{2t}. \end{aligned} \tag{10}$$

The code below defines the system of equations using two `formula` calls which are put into a `"list"`.

```
> data("Kmenta", package = "systemfit")
> eqDemand <- consump ~ price + income
> eqSupply <- consump ~ price + farmPrice + trend
> system <- list(demand = eqDemand, supply = eqSupply)
```

Two parametric models are fitted to the data using the function `systemfit`: one assuming that there is no correlation of the errors setting (the default), `OLS.fit` below; and another one assuming the existence of correlation in the system error term setting `method = "SUR"`, `FGLS1.fit` below. Arguing that the coefficients in (10) may change over time, the corresponding TVSUREs are fitted by using the the function `tvSURE` with the default in the argument `method` and by `method = "tvFGLS"`, respectively. They are denoted by `TVOLS.fit` and `TVFGLS1.fit`.

```
> OLS.fit <- systemfit::systemfit(system, data = Kmenta)
> FGLS1.fit <- systemfit::systemfit(system, data = Kmenta, method = "SUR")
> TVOLS.fit <- tvSURE(system, data = Kmenta)
> TVFGLS1.fit <- tvSURE(system, data = Kmenta, method = "tvFGLS")
```

In the previous chunk, the FGLS and TVFGLS estimators use only one iteration. However, the user can choose the iterative FGLS and the iterative TVFGLS models, which estimate the coefficients iteratively until convergence. The convergence level can be chosen with the argument `tol` (1e-05 by default) and the argument `maxiter` with the maximum number of iterations. The following chunk illustrates its usage:

```
> FGLS2.fit <- systemfit::systemfit(system, data = Kmenta, method = "SUR",
+                                   maxiter = 100)
> TVFGLS2.fit <- tvSURE(system, data = Kmenta, method = "tvFGLS",
+                       control = list(tol = 0.001, maxiter = 100))
```

Some of the coefficients can be restricted to have a certain constant value in `tvSURE`. This can aid statistical inference to test certain conditions. See an example of this below. Matrix `R` has as many rows as restrictions in `r` and as many columns as regressors in the model. In this case, Model (10) has 7 coefficients which are ordered as they appear in the list of formulas. Note that the time-varying coefficient of the variable `trend` is redundant when an intercept is included in the second equation of the TVSURE. Therefore, we want to restrict its coefficient to zero. For illustration, we also impose $\beta_{11,t} - \beta_{21,t} = 0.5$:

```
> Rrestr <- matrix(0, 2, 7)
> Rrestr[1, 7] <- 1; Rrestr[2, 2] <- 1; Rrestr[2, 5] <- -1
> qrestr <- c(0, 0.5)
> TVFGLS.rest <- tvSURE(system, data = Kmenta, method = "tvFGLS",
+                       R = Rrestr, r = qrestr,
+                       bw = TVFGLS1.fit$bw, bw.cov = TVFGLS1.fit$bw.cov)
```

Application to asset management

Several studies have argued that the three-factor model by Fama and French (1993) does not explain the whole variation in average returns. In this line, Fama and French (2015) added two new factors that measure the differences in profitability (robust and weak) and investment (conservative and aggressive), creating their five-factor model (FF5F). This model has been applied in Fama and French (2017) to analyse the international markets. A time-varying coefficients version of the FF5F has been studied in Casas et al. (2019), whose dataset is included in the **tvReg** under the name of `FF5F`. The TVFF5F model is

$$\begin{aligned} R_{it} - RF_{it} =& a_{it} + b_{it}\,(RM_{it} - RF_{it}) + s_{it}\,SMB_{it} + h_{it}\,HML_{it} \\ &+ r_{it}\,RMW_{it} + c_{it}\,CMA_{it} + u_{it}, \end{aligned} \tag{11}$$

where R_{it} refers to the price return of the asset of certain portfolio for market i at time t, RF_t is the risk free return rate, and RM_t represents the total market portfolio return. Therefore, $R_{it} - RF_{it}$ is the expected excess return and $RM_{it} - RF_{it}$ is the excess return on the market portfolio. The other factors, SMB_t stands for "small minus big" and represents the size premium, HML_t stands for "high minus low" and represents the value premium, RMW_t is a profitability factor, and CMA_t accounts for the investment capabilities of the company. Finally, the error term structure is

$$E(u_{it}u_{js}) = \begin{cases} \sigma_{iit} = \sigma_{it}^2 & i = j, \quad t = s \\ \sigma_{ijt} & i \neq j, \quad t = s \\ 0 & t \neq s. \end{cases}$$

The `FF5F` dataset has been downloaded from the Kenneth R. French (2016) data library. It contains the five factors from four different international markets: North America (NA), Japan (JP), Europe (EU), and Asia Pacific (AP). For the dependent variable, the excess returns of portfolios formed on size and book-to-market have been selected. The period runs from July 1990 to August 2016 and it has a monthly frequency. The data contains the Small/Low, Small/High, Big/Low and Big/High portfolios. The factors in the TVFF5F model explain the variation in returns well if the intercept is statistically zero. The lines of code below illustrate how to fit a TVSURE to the Small/Low portfolio.

```
> data("FF5F")
> eqNA <- NA.SMALL.LoBM - NA.RF ~ NA.Mkt.RF + NA.SMB + NA.HML + NA.RMW + NA.CMA
> eqJP <- JP.SMALL.LoBM - JP.RF ~ JP.Mkt.RF + JP.SMB + JP.HML + JP.RMW + JP.CMA
> eqAP <- AP.SMALL.LoBM - AP.RF ~ AP.Mkt.RF + AP.SMB + AP.HML + AP.RMW + AP.CMA
> eqEU <- EU.SMALL.LoBM - EU.RF ~ EU.Mkt.RF + EU.SMB + EU.HML + EU.RMW + EU.CMA
> system2 <- list(NorthA = eqNA, JP = eqJP, AP = eqAP, EU = eqEU)
```

```
> TVFF5F <- tvSURE(system2, data = FF5F, method = "tvFGLS",
+                  bw = c(0.56, 0.27, 0.43, 0.18), bw.cov = 0.12)
```

The package **tvReg** also includes the functionality to compute confidence intervals for the coefficients of class attributes `"tvlm"`, `"tvar"`, `"tvplm"`, `"tvsure"` and `"tvirf"` by extending the `confint` method. The algorithm in Fan and Zhang (2000) and Chen et al. (2017) to calculate bootstrap confidence intervals has been adapted for all these class attributes. Argument `level` is set to 0.95 (95% confidence interval) by default. Argument `runs` (100 by default) is the number of resamples used in the bootstrapping calculation. Note that the calculation using `runs = 100` can take long, so we suggest to try a small value in `runs` first to get an initial intuition of the results. Because coefficients are time-varying, only wild bootstrap residual resampling is implemented. Two choices of wildbootstrap are allowed in argument `tboot`: the default one proposed in Mammen (1993) (`tboot = "wild"`); and the standard normal (`tboot = "wild2"`).

In the backend code, coefficient estimates from all replications are stored in the `BOOT` variable. In this way, calculations are not done again if the user chooses a different `level` for the same object. In the chunk below, the `confint` method calculates the 90% confidence interval of the object `TVFF5F`. Posteriorly, the 95% interval is calculated quickly because the resample calculations in the first interval are re-used for the second.Thus, the 90% confidence interval calculation takes around 318 seconds with a 2.2 GHz Intel Core i7 processor and the posterior 95% confidence interval takes only around 0.7 seconds.

```
> TVFF5F.90 <- confint(TVFF5F, level = 0.90)
> TVFF5F.95 <- confint(TVFF5F.90)
```

The `plot` method is implemented for each of the six class attributes in **tvReg**. For example, the 95% confidence intervals of the intercept for the North American, Japanese, Asia Pacific and European markets Figure 1 are with `plot` statement below, that produces four independent plots of the first variable (the intercept in this case) in each equation due to argument `vars = 1`.

```
> plot(TVFF5F.95, vars = 1)
```

The user can also choose to plot the coefficients of several variables and/or equations. Plots will be grouped by equation, with a maximum of three variables per plot. The piece of code below show how to plot the coefficients of the second and third variables from the Japan market equation, which results can be seen in Figure 2.

```
> plot(TVFF5F.95, vars = c(2, 3), eqs = 2)
```

4 Standard usage of `tvPLM`

The `tvPLM` method is inspired by the `plm` method from the package **plm**. It converts `data` into an object of the class attribute `"pdata.frame"` using argument `index` to define the cross-section and time dimensions. If `index = NULL` (default), the two first columns of `data` define the dimensions. The `tvPLM` wraps the `tvRE` and `tvFE` methods to estimate the coefficients of time-varying panel data models.

The user can provide additional optional arguments to modify the default estimation. See section 2.3 for details on arguments `z`, `bw`, `est` and `tkernel`. Furthermore, argument `method` defines the estimator used. The possible choices based on package **plm** choices are: `"pooling"` (default), `"random"` and `"within"`.

Application to health policy

The income elasticity of healthcare expenditure is defined as the percentage change in healthcare expenditure in response to the percentage change in income per capita. If this elasticity is greater than one, then healthcare expenditure grows faster than income, as luxury goods do, and is driven by market forces alone. The heterogeneity of health systems among countries and time periods have motivated the use of panel data models, for example in Gerdtham et al. (1992) who use a FE model. Recently, Casas et al. (2021) have investigated the problem from the time-varying panel models perspective using the TVFE estimation. In addition to the income per capita, measured by the log GDP, the authors use the proportion of population over 65 years old, the proportion of population under 15 years old and the share of public funding of healthcare. The income elasticity estimate with a FE implemented in the `plm` is greater than 1, a counter-intuitive result. This issue is resolved using the TVFE implemented in the **tvReg**. The code below estimates coefficients with the parametric and semiparametric models:

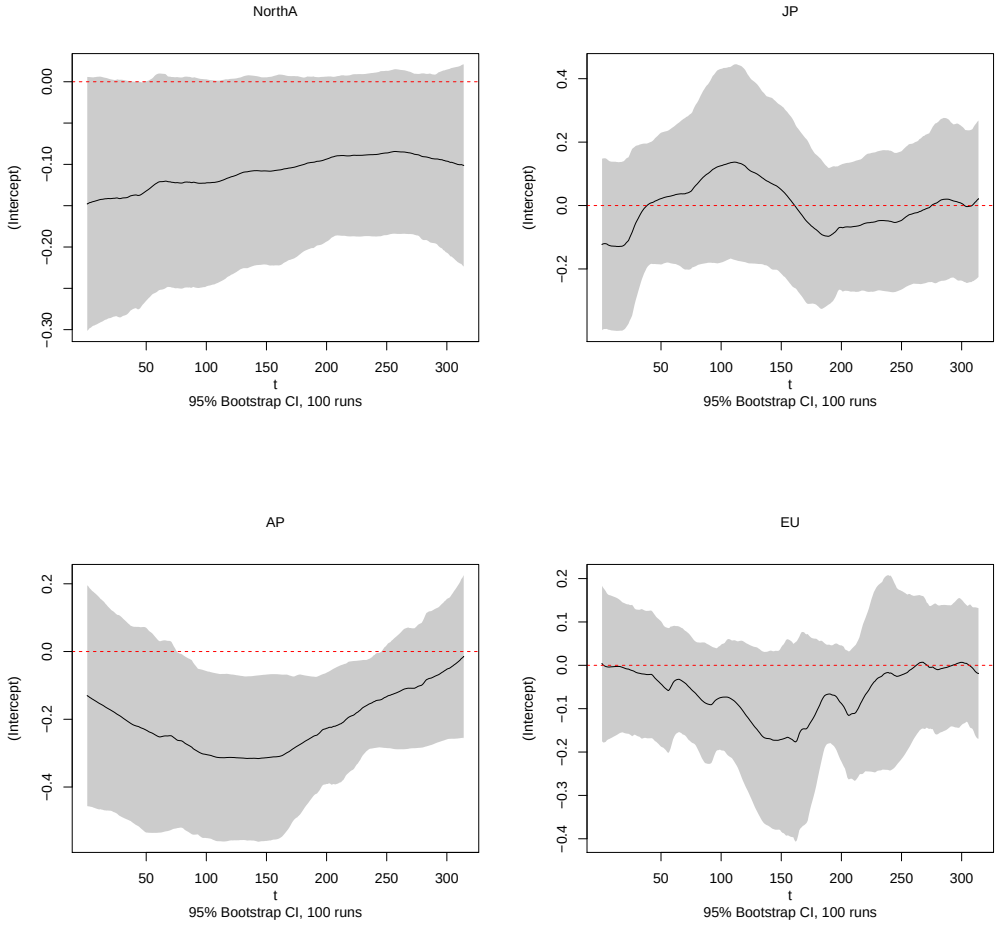


Figure 1: Intercept estimates of a Small/Low portfolio in the four markets (left to right, top to bottom: North America, Japan, Asia Pacific and Europe). The solid lines indicate the estimates, the grey bands are their 95% bootstrap confidence intervals and the red dashed lines indicate zero. Only the Asia Pacific market intercepts are statistically different from zero during a large period, implying that the FF5F does not explain excess returns well for the Asia Pacific market.

```
> data("OECD")
> elast.fe <- plm::plm(lhe ~ lgdp + pop65 + pop14 + public, data = OECD,
+                      index = c("country", "year"), model = "within")
> elast.tvfe <- tvPLM (lhe ~ lgdp + pop65 + pop14 + public, data = OECD,
+                      index = c("country", "year"), method = "within",
+                      bw = 0.67)
> elast.fe <- confint(elast.fe)
> elast.tvfe <- confint(elast.tvfe)
```

Figure 3 shows the elasticity estimates using the FE and TVFE estimators. The constant coefficients model (dashed line) suggests that healthcare is a luxury good (over 1), while the time-varying coefficients (solid line) model suggests it is a value under 0.8.

5 Standard usage of `tvVAR` and `tvIRF`

A TVVAR(p) model is a system of time-varying autoregressive equations of order p. The dependent variable, `y`, is of the class attribute `"matrix"` or `"data.frame"` with as many columns as equations. Regressors are the same for all equations and they contain an intercept if the argument `type = "const"` (default) or not if `type = "none"`; lagged values of `y`; and other exogenous variables in `exogen`. Econometrically, the `tvOLS` method is called to calculate the estimates for each equation independently using one bandwidth per equation. The user can choose between automatic bandwidth selection; or

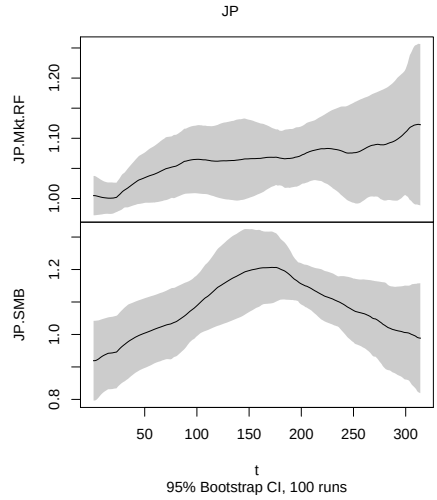


Figure 2: Coefficient estimates of excess returns on the market portfolio (JP.Mkt.RF) and JP.SMB factors for a Small/Low portfolio in the Japan market. The solid line indicates the estimates and the grey bands are their 95% bootstrap confidence intervals. It seems that the effect of the market return over the asset return increases slightly over time, while the effect of the size premium over the asset return has an inverted U shape over the time period.

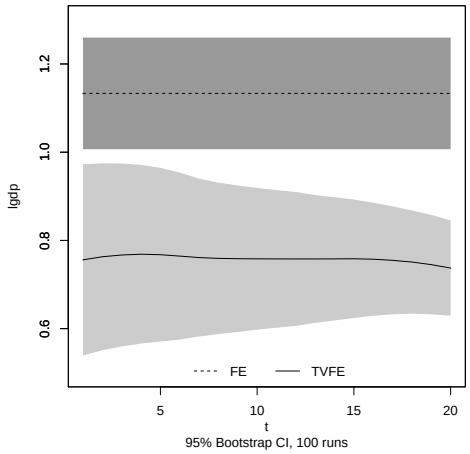


Figure 3: Comparison of income elasticity of healthcare expenditure in OECD countries. The dashed line with a dark grey band corresponds to the FE estimate and its 95% bootstrap confidence interval, while the solid line with a light band corresponds to the TVFE estimates and their 95% confidence intervals. There is a clear difference in the elasticity estimates of the two models.

entering a one value in `bw`, meaning that all equations will be estimated with the same bandwidth; or a vector of bandwidths, one for each equation. The `tvVAR` returns a `list` of the class attribute `tvvar`, which can be used to estimate the TVIRF model with the function `tvIRF`.

Application to monetary policy

The assessment and forecast of the effects of monetary policy on macroeconomic variables, such as inflation, economic output and employment is commonly modelled using the econometric framework of VAR and interpreted by the IRF. In recent years, scholars of macroeconometrics have searched intensely for a way to include time variation in the coefficients and covariance matrix of the VAR model. The reason for this is that the macroeconomic climate evolves over time and effects of monetary policy must be identified locally rather than globally. In the Bayesian framework, Primiceri (2005) used the Carter and Kohn (1994) algorithm to fit the TVP-VAR to this monetary policy problem. Results of the latter can be replicated with the functions in the package **bvarsv** and compared with results in the

tvReg that fits the following TVVAR(4):

$$\text{inf}_t = a_t^1 + \sum_{i=1}^{4} b_{it}^1 \, \text{inf}_{t-i} + \sum_{i=1}^{4} c_{it}^1 \, \text{une}_{t-i} + \sum_{i=1}^{4} d_{it}^1 \, \text{tbi}_{t-i} + u_t^1$$

$$\text{une}_t = a_t^2 + \sum_{i=1}^{4} b_{it}^2 \, \text{inf}_{t-i} + \sum_{i=1}^{4} c_{it}^2 \, \text{une}_{t-i} + \sum_{i=1}^{4} d_{it}^2 \, \text{tbi}_{t-i} + u_t^2$$

$$\text{tbi}_t = a_t^3 + \sum_{i=1}^{4} b_{it}^3 \, \text{inf}_{t-i} + \sum_{i=1}^{4} c_{it}^3 \, \text{une}_{t-i} + \sum_{i=1}^{4} d_{it}^3 \, \text{tbi}_{t-i} + u_t^3.$$

Central banks commonly regulate the money supply by changing the interest rates to keep a stable inflation growth. The R code below uses macroeconomic data from the United States, exactly the one used in Primiceri (2005), with the following three variables: inflation rate (`inf`), unemployment rate (`une`) and the three months treasury bill interest rate (`tbi`). For illustration, a VAR(4) model is estimated using the function `VAR` from the package **vars**, a TVVAR(4) model is estimated using the function `tvVAR` from the package **tvReg** and a TVP-VAR(4) model is estimated using the function `bvar.sv.tvp` from the package **bvarsv**. Furthermore, their corresponding impulse response functions with horizon 20 are calculated to forecast how the inflation responds to a positive shock in interest rates. The TVVAR(4) can also be estimated with function `tvmvar` from R package **mgm**, which will give the same coefficient estimates than the `tvVAR` for the Gaussian kernel and same bandwidth. However, package **mgm** does not have an impulse response function and, for this reason, it is left out of the example.

```
> data(usmacro, package = "bvarsv")
> VAR.usmacro <- vars::VAR(usmacro, p = 4, type = "const")
> TVVAR.usmacro <- tvVAR(usmacro, p = 4, bw = c(1.14, 20, 20), type = "const")
> TVPVAR.usmacro <- bvarsv::bvar.sv.tvp(usmacro, p = 4, pdrift = TRUE, nrep = 1000,
+                                       nburn = 1000, save.parameters = TRUE)
```

The user can provide additional optional arguments to modify the default estimation. See Section Standard usage of `tvSURE` to understand the usage of arguments `bw`, `tkernel`, `est` and `singular.ok`. In addition, the function `tvVAR` has the following arguments:

Number of lags
The number of lags is given by the model order set in the argument `p`.

Exogen variables
Other exogenous variables can be included in the model using the argument `exogen`, which accepts a vector or a matrix with the same number of rows as the argument `y`.

Type
The default model contains an intercept (i.e., it has a mean different from zero). The user can set argument `type = "none"`, so the model has mean zero.

The variance-covariance matrix from the residuals of a TVVAR(p) can be used to calculate the orthogonal TVIRF. The `plot` method for object of class attribute `"tvvar"` displays as many plots as equations, each plot with the fitted and residuals values as it is shown in Figure 4 obtained with:

```
> plot(TVVAR.usmacro)
```

Figure 4 shows the residuals of the inflation equation that has a mean close to zero and the fitted values are fitting the observed values closely.

Function `tvIRF` estimates the TVIRF with main argument, `x`, which is an object of class attribute `"tvvar"` returned by the function `tvVAR`. The user can provide additional optional arguments to modify the default estimation as explained below.

Impulse and response variables
The user has the option to pick a subset of impulse variables and/or response variables using arguments `impulse` and `response`.

Horizon
The horizon of the TVIRF coefficients can be chosen by the user with argument `n.ahead`, the default is 10.

Orthogonal TVIRF
The orthogonalised impulse response function is computed by default (`ortho = TRUE`). In the orthogonal case, the estimation of the variance-covariance matrix of the errors is estimated as

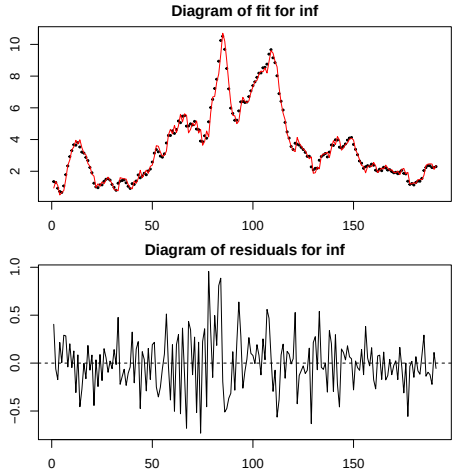


Figure 4: Returns and fitted values of object `TVVAR.usmacro` for the inflation equation. The dots in the top plot represent the observed values and the red line represents the fitted values, while the black line in the bottom plot represents the returns of the estimation. The model fits the observed values well and the returns appear to have zero mean and constant variance.

time-varying (`ortho.cov = "tv"`) by default (see Section Estimating a time-varying variance-covariance matrix for theoretical details). Note that the user can enter a value of the bandwidth for the variance-covariance matrix estimation in `bw.cov`. It is possible to use a constant variance-covariance matrix by setting `ortho.cov = "const"`.

Cumulative TVIRF

If the user desires to obtain the cumulative TVIRF values, then argument `cumulative` must be set to `TRUE`.

Following the previous example, the lines of code below estimate the IRF using the package **vars**, the TVP-IRF using the package **bvarsv** and the TVIRF using the package **tvReg**.

```
> IRF.usmacro <- vars::irf(VAR.usmacro, impulse = "tbi", response = "inf", n.ahead = 20)
> TVIRF.usmacro <- tvIRF(TVVAR.usmacro, impulse = "tbi", response = "inf", n.ahead = 20)
> TVPIRF.usmacro <- bvarsv::impulse.responses(TVPVAR.usmacro, impulse.variable = 3,
+                                             response.variable = 1, draw.plot = FALSE)
```

A comparison of impulse response functions from the three estimations is plotted in Figure 5, whose R code is shown below:

```
> irf1 <- IRF.usmacro$irf[["tbi"]]
> irf2 <- TVIRF.usmacro$irf[["tbi"]]
> irf3 <- TVPIRF.usmacro$irf
> ylim <- range(irf1, irf2[150,,], irf3[50,])
> plot(1:20, irf1[-1], ylim = ylim, main = "Impulse variable: tbi from 1990Q2",
+      xlab ="horizon", ylab ="inf", type ="l", lwd = 2)
> lines(1:20, irf2[150,,-1], lty = 2, lwd = 2)
> lines(1:20, irf3[50,], lty = 3, lwd = 2)
```

Figure 5 displays the IRF, the TVIRF and the TVP-IRF (the two latter at time 150 in our database, which corresponds to the second quarter of 1990) for horizons 1 to 20. The IRF and TVIRF follow a similar pattern: a positive shock of one unit in the short-term interest rates (`tbi`) during 1990Q2 results in an initial drop in inflation during the first three months, followed by an increase for two or three months and finally in a steady decrease until it plateaus one year after. The left plot shows an increase in inflation during the first three months and a drop after.

The `confint` method is also implemented for the class attribute `"tvirf"`. Remember that the TVIRF model contains one impulse response function for each data time record. So, the full plot of TVIRF would have as many lines as the number of rows in the dataset. Instead, the `plot` method displays only one line by default, the mean value of all those impulse response functions and it issues a warning. The user can enter one or several values into argument `obs.index` to plot the IRF at the desire point(s) in time.

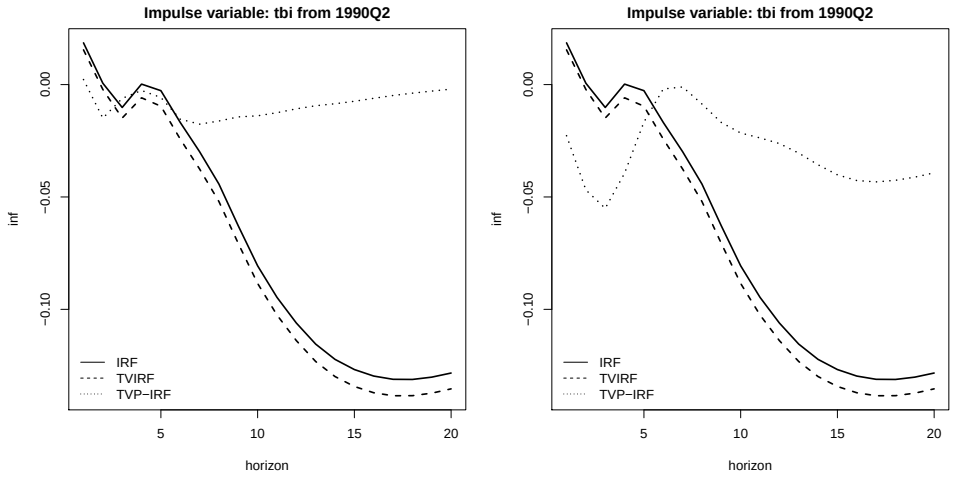


Figure 5: Estimated response of inflation (inf) to an increase in interest rates (tbi) of one unit during 1990Q2.The dashed line correspond to the IRF estimates, the solid line to the TVIRF and the dotted line to the Bayesian estimates. It appears that the Bayesian estimates are very different from those of the other two models.

6 Estimating a time-varying variance-covariance matrix

The time-varying variance-covariance matrix of two or more series is estimated nonparametrically in **tvReg**. Given a random process $y_i = (y_{i1}, \ldots, y_{iT})^\top$, such that $E(y_{it}) = 0$ and $E(y_{it}y_{i't'}) = \sigma_{ii't}$ if $t = t'$ and zero otherwise. Thus, the variance-covariance matrix for time t is denoted by Σ_t with elements $\sigma_{ii',t}$ with $1 \leq i, i' \leq N$. Given that Σ_t is locally stationary, its local linear estimator is defined by

$$\text{vech}(\tilde{\Sigma}_\tau) = \sum_{t=1}^{T} \text{vech}(y_t^\top y_t) K_h(t-\tau) \frac{s_2 - s_1(\tau - t)}{s_0 s_2 - s_1^2} \tag{12}$$

where $s_j = \sum_{t=1}^{T} (\tau - t)^j K_b(\tau - t)$ for j = 0, 1, 2. As shown previously, $K_b(\cdot)$ is a symmetric kernel function heavily concentrated around the origin, $\tau = t/T$ is the focal point and b is the bandwidth parameter. Note that a single bandwidth is used for all co-movements, which ensures that $\tilde{\Sigma}_\tau$ is positive definite.

The user must be aware that the local linear estimator can return non-positive definite matrices for small samples. Although the local constant estimator, calculated when $s_1 = s_2 = 1$ in (12), does not have as good asymptotic properties in the boundaries as the local linear estimator, it always provides positive definite matrices, which is a desirable property of an estimator of a variance-covariance matrix. Therefore, it is the default estimator in the function `tvCov`.

The function `tvCov` is called by the function `tvIRF` to calculate the orthogonal TVIRF, and by the function `tvSURE` for `method = "tvFGLS"` to estimate the variance-covariance matrix of the error term. The function `tvCov` can generally be used to estimate the time-varying covariance matrix of any two or more series.

Application to portfolio management

Aslanidis and Casas (2013) consider a portfolio of daily US dollar exchange rates of the Australian dollar (AUS), Swiss franc (CHF), euro (EUR), British pound (GBP), South African rand (RAND), Brazilian real (REALB) and Japanese yen (YEN), over the period from January 6, 1999 until May 7, 2010 (T = 2855 observations). This dataset contains the standarised rates after "devolatilisation"; i.e., after standarising the rates using the GARCH(1,1) estimates of the volatility and it is available in the **tvReg** under the names of CEES. A portfolio consisting of these currencies is well diversified containing some safe haven currencies, active and liquid currencies and currencies that perform well in times of high interest rates. The estimation of the correlation matrix among these currencies is essential for portfolio management. The model is

$$\begin{aligned} r_{p,t} &= \omega_t^\top r_t \\ h_{p,t} &= \omega_t^\top H_t \omega_t \end{aligned}$$

where $r_{p,t}$ and $h_{p,t}$ are the return and variance of the portfolio at time t. Variable ω_t is a vector with the weight of each currency in the portfolio strategy at time t. The portfolio variance-covariance matrix is denoted by H_t, and it may vary with time for a dynamic investment strategy. This matrix can be estimated using the function `tvCov` and then used in risk management, for example to calculate the Value-at-risk, denoted by VaR in the financial literature. The VaR, not be confused with the VAR, measures the level of financial risk of a portfolio, asset or firm. The VaR of an asset X, with distribution function F_X, at the confidence level α is defined as $\text{VaR}_\alpha = \inf\{x : F_X(x) > \alpha\}$. Commonly, the distribution function of X is assumed to be Gaussian with unknown variance. In a portfolio framework, the variance-covariance matrix is estimated to calculate the VaR of a portfolio together with the portfolio weights (`omega` in the code below). The portfolio weights are the percentage of the total portfolio investment in each asset and can be chosen to be constant or changing over time. In the code below, weights are calculated by minimum variance at each point in time. The estimated VaR of this example portfolio is shown in Figure 6.

```
> data(CEES)
> VaR <- numeric(nrow(CEES))
> Ht <- tvCov(CEES[, -1], bw = 0.12)
> e <- rep (1, ncol(CEES)-1)
> for (t in 1:nrow(CEES)){
+   omega <- solve(Ht[,,t])%*%e/((t(e)%*%solve(Ht[,,t])%*%e)[1])
+   VaR[t] <- abs(qnorm(0.05))*sqrt(max(t(omega)%*%Ht[,,t]%*%omega,0))
+ }
> plot(as.Date(CEES[, "Date"]), VaR, type ="l", xlab = "year",
+      ylab = expression(VaR[t]), main="VaR of CEES over time")
```

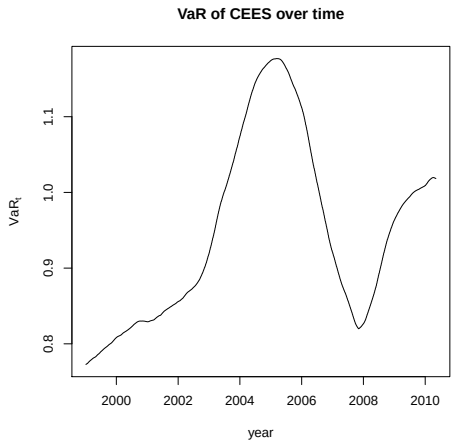


Figure 6: Dynamics of the Value-at-risk of the CEES exchange rates portfolio over time. The solid line represents the VaR. It appears that the risk of potential financial losses of this portfolio increased up to year 2005, decreasing then until 2008 and turn up again afterwards.

7 Single-equation linear models with time-varying coefficients

A varying coefficients linear model (TVLM) is generally expressed by

$$y_t = x_t^\top \beta(z_t) + u_t, \;\; t = 1, \ldots, T, \tag{13}$$

where y_t is the response or dependent variable, $x_t = (x_{1t}, x_{2t}, \ldots, x_{dt})^\top$ is a vector of regressors at time t, $\beta(z_t)$ is the vector of coefficients at time t and u_t is the error term which satisfies $E(u_t|x_t) = 0$ and $E(u_t^2|x_t) = \sigma^2$. There are not enough degrees of freedom in the TVLM for a meaningful OLS estimation, but it may be estimated with the TVOLS displayed in Equation 2. The particular case of $x_t = (y_{t-1}, y_{t-2}, \ldots, y_{t-p})$ corresponds to the time-varying autoregressive model, TVAR(p), which is also estimated with the TVOLS.

The case of $z_t = \tau = t/T$ was firstly studied in Robinson (1989) for stationary processes and generalised to nonstationary processes and correlated errors by Chang and Martinez-Chombo (2003) and Cai (2007) among others. Recently, Chen et al. (2017) apply it to the Heterogeneous Auto-Regressive (HAR) model of Corsi (2009) for the realized volatility of S&P 500 index returns. It is a very

flexible approach, but forecasts are not consistent because there is no information from the dependent variable at time $T+1$. On the other hand, the case of a random z_t has been studied for iid or stationary processes by Hastie and Tibshirani (1993) and Cai et al. (2000); and nonstationary regressors or/and nonstationary z_t have been studied by Chang and Martinez-Chombo (2003), Cai et al. (2009), Zhang and Wu (2012), Sun et al. (2013) and Gao and Phillips (2013). Das (2005) and Xiao (2009) have used the approach for instrumental variables and cointegration. In summary, this estimator is consistent and asymptotically normal for several types of dependency of $\{(x_t, z_t, u_t)\}$.

8 Standard usage of `tvLM`

The function `tvLM` fits a TVLM using the `tvOLS` method. The `tvLM` follows the standards of the function `lm` with main arguments `formula` and `data`. The only mandatory argument is `formula`, which should be a single formula for a single-equation model. This arguments follows the standard regression `formula` in R. The function `tvLM` returns an object of the class attribute `tvlm`. This model is in some cases a GAM-type model which is implemented in the comprehensive and well-established **mgcv** package. The **mgcv** uses a methodology different from kernel smoothing to estimate the varying coefficients, involving splines and quasi-maximum likelihood estimation. The advantage of using kernel smoothing techniques to estimate the TVLM is that it can handle dependency and any kind of distribution in the error term. For illustration of this difference between the two packages in relation to the TVLM, the following model is generated:

$$y_t = \beta_{1t} x_{1t} + \beta_{2t} x_{2t} + u_t, \;\; t = 1, \ldots, T, \tag{14}$$

where $\beta_{1t} = \sin(2\pi\tau)$ and $\beta_{2t} = 2\tau$ with $\tau = t/T$ and $T = 1000$. The regressors, $x_{1t} \sim t_2$ (symmetric) and $x_{2t} \sim \chi^2_4$, are independent of the error term, $u_t \sim \chi^2_2$ which has an exponential dependency in the covariance matrix given by $Cov(u_t, u_{t+h}) = e^{-|h|/10}$ and does not follow an exponential-family distribution. The LM, TVLM and GAM models are fitted to the data. The process generation and the fitting of a classical LM, a TVLM and a GAM are shown in the following chunk. Figure 7 compares the different estimates with the true β_{1t}, β_{2t}. As expected, the estimates from `lm` are constant and lie around the average of all β_{1t} and β_{2t}, while the estimates of `tvLM` and `gam` follow the dynamics of the varying coefficients. Besides the estimates of `gam` fit β_{1t} well, but not β_{2t} although the latter is a simple linear function. This issue is caused by the autocorrelated error term with a non-exponential distribution. On the other hand, the `tvLM`, although it requires for a longer computation time, it is able to fit both coefficients well.

```
> tau <- seq(1:1000)/1000
> d <- data.frame(tau, beta1 = sin(2 * pi * tau), beta2 = 2 * tau,
+                 x1 = rt(1000, df = 2), x2 = rchisq(1000, df = 4))
> error.cov <- exp(-as.matrix(dist(tau))/10)
> error <-  t(chol(error.cov)) %*% rchisq(N, df = 2)
> d <- transform(d, y = x1 * beta1 + x2 * beta2 + error)
> lm1 <- stats::lm(y ~ x1 + x2, data = d)
> TVLM1 <- tvLM(y ~ x1 + x2,  data = d, bw = 0.05, est ="ll")
> gam1 <- mgcv::gam(y ~ s(tau, by = x1) + s(tau, by = x2), data = d)
```

In addition to `formula`, the function `tvLM` has the arguments described in Section Standard usage of `tvSURE` above. Also methods `confint`, `fitted`, `print`, `plot`, `residuals` and `summary` are implemented for class `"tvlm"`.

The `summary` method displays: (i) a summary of all coefficient values over the whole time period, (ii) the value of the bandwidth(s), and (iii) a measure of goodness-of-fit, pseudo-R^2. The latter is printed for the class attributes `"tvsure"`, `"tvplm"`, `"tvvar"`, `"tvlm"` and `"tvar"` and it is calculated with the classical equation,

$$R^2 = 1 - \frac{\sum_{t=1}^{T}(y_t - \hat{y})^2}{\sum_{t=1}^{T}(y_t - \bar{y})^2},$$

where y_t is the dependent variable, $\bar{y}$ is its mean and $\hat{y}_t$ are the fitted values. For multiple equation models, one pseudo-R^2 is calculated for each equation.

9 Standard usage of `tvAR`

A TVAR model is a particular case of TVLM whose regressors contain lagged values of the dependent variable, `y`. The number of lags is given by the model order set in the argument `p`. Other exogenous

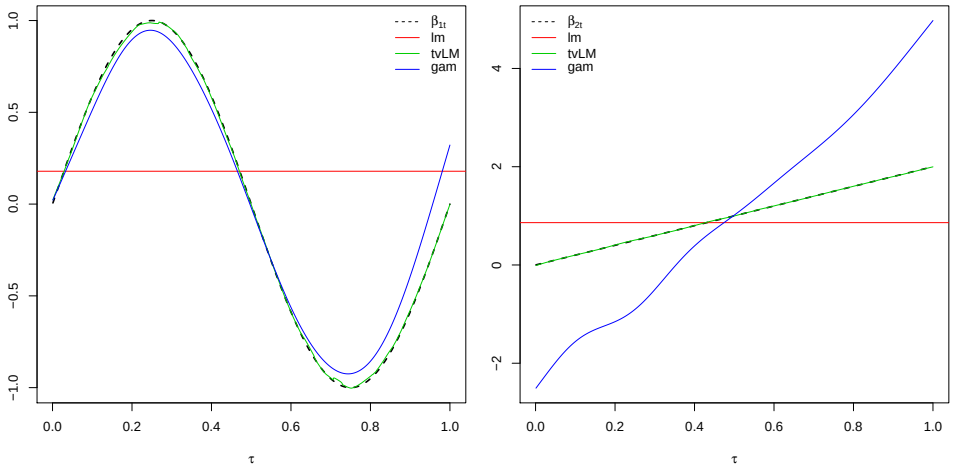


Figure 7: Comparison of the `lm`, `tvLM` and `gam` estimates of β_{1t} and β_{2t}. The true values are plotted in black, the red lines represent the `lm` estimates, the green lines refer to the `tvLM` estimates and the blue lines represent the `gam` estimates. This result suggests that the TVLM is preferable for modelling non-linear varying coefficients under strong dependency.

variables can be included in the model using the argument `exogen`, which accepts a vector or a matrix with the same number of rows as the argument `y`. An intercept is included by default unless the user enters `type = "none"` into the function call. Econometrically, this function also wraps the `tvOLS` estimator, which needs a bandwidth `bw` that is automatically selected when the user does not enter any number. An object of the class attribute `"tvar"` is returned by the function `tvAR`.

The user can provide additional optional arguments to modify the default estimation of the function `tvAR`. See Section Standard usage of `tvSURE` to understand the usage of arguments `bw`, `tkernel`, `est` and `singular.ok` and Section Standard usage of `tvVAR` and `tvIRF` to understand the usage of argument `type`. In addition, the function `tvAR` has the following argument:

Coefficient restrictions

An autoregressive process of order p does not necessarily contain all the previous p lags of y_t. Argument `fixed`, with the same format as in the function `arima` from the package **stats**, permits to impose these restrictions. The order of variables in the model is: intercept (if any), lag 1, lag 2, ..., lag p and exogenous variable (if any). By default, the argument `fixed` is a vector of NAs with length the number of coefficients in the model. The user can enter a vector in the argument `fixed` with zeros in the positions corresponding to the restricted coefficients.

Application to risk management

The realized variance (RV) model was popularised in the financial literature by Andersen and Bollerslev (1998), who show that the use of intraday data can offer an accurate forecast of daily variance. It is defined as $RV_t = \sum_{i=1}^{N} r_{it}^2$, where r_{it} is the price return at minute i of day t. The autocorrelation function of the RV also shows signs of long memory in the process, which can be accounted for by the heterogeneous RV (HAR) model of Corsi (2009):

$$RV_t = \beta_0 + \beta_1 RV_{t-1} + \beta_2 RV_{t-1|t-5} + \beta_3 RV_{t-1|t-22} + u_t. \tag{15}$$

Here, $RV_{t-1|t-k} = \frac{1}{k}\sum_{i=1}^{j} RV_{t-i}$. In this model, the current RV_t depends on its immediately previous value, RV_{t-1}, its medium-term memory factor, $RV_{t-1|t-5}$ and its long-term memory factor, $RV_{t-1|t-22}$. Basically, the HAR model may be seen as an AR(1) model with two exogenous variables.

It is likely that changes in the business cycles affect the coefficients in (15). Chen et al. (2017) coined the time-varying coefficient HAR, whose coefficients are functions of the rescaled time period. The `RV` dataset contains daily variables running from January 3, 1990 until December 19, 2007 that have been computed from 5 minute intraday data from Store (2017). This period coincides with the period in Bollerslev et al. (2009). The variable names in this dataset are `RV`, `RV_lag`, `RV_week`, `RV_month` and `RQ_lag_sqrt` and correspond to the RV_t, RV_{t-1}, $RV_{t-1|t-5}$, $RV_{t-1|t-22}$ and $RQ_{t-1}^{1/2}$ in Model (15).

```
> data("RV")
> RV2 <- head(RV, 2000)
```

```
> HAR <- with(RV2, arima(RV, order = c(1, 0, 0), xreg = cbind(RV_week, RV_month)))
> TVHAR<- with(RV2, tvAR(RV, p = 1, bw = 0.8, exogen = cbind(RV_week, RV_month)))
```

Bollerslev et al. (2016) extended the Model (15) to control for the effect of the realized quarticity (RQ) on the relationship between the future RV and its near past values. They present the HARQ model,

$$RV_t = \beta_0 + (\beta_1 + \beta_{1Q} RQ_{t-1}^{1/2})\, RV_{t-1} + \beta_2\, RV_{t-1|t-5} + \beta_3\, RV_{t-1|t-22} + u_t. \tag{16}$$

The HARQ model is a HAR model whose RV_{t-1} term's coefficient is a linear function of the squared root of RQ at time $t-1$. The RQ changes over time and it will be larger during periods of more uncertainty. Casas et al. (2018) appreciated that the variation of this coefficient may not be linear and proposed the TVHARQ model,

$$RV_t = \beta_0(z_t) + \beta_1(z_t)\, RV_{t-1} + \beta_2(z_t)\, RV_{t-1|t-5} + \beta_3(z_t)\, RV_{t-1|t-22} + u_t, \tag{17}$$

where the smoothing variable, $z_t = RQ_{t-1}^{1/2}$. This model is a TVAR(1) process and can be estimated with the function `tvAR` or with the function `tvLM` as it is shown in the chunk below.

```
> HARQ <- with(RV2, lm(RV ~ RV_lag + I(RV_lag*RQ_lag_sqrt) + RV_week + RV_month))
> TVHARQ <- with(RV2, tvAR(RV, p = 1, exogen = cbind(RV_week, RV_month),
+                          z = RQ_lag_sqrt, cv.block = 10))
```

10 Prediction and forecast in time-varying coefficient models

Estimation is a useful tool to understand the patterns and processes hidden in known data. Prediction and forecast are the mechanisms to extend this understanding to unknown data. Although the two terms are often used indistinctively, the term prediction is broader than the term forecast which is reserved for time-series models and consists on using historical data to infer the future. For example, we speak of predicting values from a linear regression fitted to cross-sectional data and of forecasting future values from an AR(p) model.

The *prediction* of the dependent variable at time $T+h$ (horizon of length h) in a linear regression is $\hat{y}_{T+h} = x_{T+h}^\top \hat{\beta}$ for $h \geq 1$. Future values, x_{T+h}, must be known to calculate the prediction. In time series, the prediction of future values has a slightly different nature and then is when we use the word *forecast*. The regressors in the 1-step-ahead forecast are known, but they are effectively unknown for for longer horizons and must be forecasted first. For example, given $y_t = 5 - 0.5 y_{t-1} + u_t$ for $t = 1, \ldots, T$; the 1-step-ahead forecast is $\hat{y}^*_{T+1} = 5 - 0.5 y_T$ with known y_T. However, the 2-step-ahead forecast is $\hat{y}^*_{T+2} = 5 - 0.5 \hat{y}^*_{T+1}$, which uses the previous forecast value, $\hat{y}^*_{T+1}$.

In the **tvReg**, we refer to prediction when z_t is a random variable and to forecast when $z_t = \tau$. Note that future values of the conditional variable, z_{T+h}, must be given for prediction. For example, the prediction problem $\hat{y}_{T+h} = x_{T+h}^\top \hat{\beta}(z_t)$ for $h \geq 1$ requires the future values x_{T+h} and z_{T+h}. Whereas, the forecast problem $\hat{y}^*_{T+h} = x_{T+h}^\top \hat{\beta}(T+h)$ requires only the future values x_{T+h}. Thus, the `predict` and `forecast` methods in **tvReg** are slightly different.

11 Standard usage of predict and forecast

The `forecast` method is implemented for the class attributes `"tvsure"`, `"tvplm"`, `"tvar"`, `"tvlm"` and `"tvar"`. As an example, the three days ahead forecast of model `TVHAR`, evaluated in Section Application to risk management using the first 2000 values of the dataset `RV`, is provided in the lines of code below. This is a TVAR(1) model with two exogenous variables, `RV_week` and `RV_month`. The argument `newexogen` requires three values of these exogenous variables and variable `n.ahead = 3`.

```
> newexogen <- cbind(RV$RV_week[2001:2003], RV$RV_month[2001:2003])
> forecast(TVHAR, n.ahead = 3, newexogen = newexogen)

[1] 2.200921e-05 2.566854e-05 2.466637e-05
```

The `forecast` method requires the argument `object`. In addition, other arguments are necessary, some of them depending on the class attribute of `object`.

Forecast horizon The argument `n.ahead` is a scalar with the forecast horizon. By default, it is set to 1.

Type of forecast It is possible to run either an increasing window forecast (default), when the argument `winsize = 0` or a rolling window forecast with a window size defined in the argument `winsize`.

newdata
These arguments belong to the `forecast` methods and it is a `"vector"`, `"data.frame"` or `"matrix"` containing the new values of the regressors in the model. It is not necessary to enter the intercept. Note that this `newdata` does not refer to the variables in `exogen` which might be part of the `"tvar"` and `"tvvar"` objects. Those must be included in `newexogen`, if needed.

newexogen
This argument appears in the `forecast` method for the class attributes `"tvar"` and `"tvvar"` and it must be entered when the initial model contains `exogen` variables. It is a `"vector"`, `"data.frame"` or `"matrix"`.

The `predict` method is implemented for the same class attributes than the `forecast`. It does not require arguments `n.ahead` and `winsize`, but arguments `newdata` and `newexogen` are defined as in `forecast`. In addition, new values of the smoothing variable must be entered into the argument `newz`. This must be of the class attribute `"vector"` or `"numeric"`. The code below, predicts three future values of the `TVHAR` model fitted above.

```
> newdata <- RV$RV_lag[2001:2003]
> newexogen <- cbind(RV$RV_week[2001:2003], RV$RV_month[2001:2003])
> newz <- RV$RQ_lag_sqrt[2001:2003]
> predict(TVHARQ, newdata, newz, newexogen = newexogen)

[1] 1.741663e-05 2.402516e-05 2.088794e-05
```

The example below shows the usage of the `forecast` and `predict` methods for the class attribute `"tvsure"`.

The lines of code below forecast three values for model `TVOLS.fit` evaluated in Section Standard usage of `tvSURE`. The method needs a set of new values in the argument `newdata`, which must have the same number of columns as the original dataset.

```
> newKmenta <- data.frame(consump = c(95, 100, 102), price = c(90, 100, 103),
+                         farmPrice = c(70, 95, 103), income = c(82, 94, 115),
+                         trend = c(21:23))
> forecast(TVOLS.fit, newdata = newKmenta, n.ahead = 3)

        demand    supply
[1,]  97.92300  95.32852
[2,]  98.94076 103.48589
[3,] 105.36951 106.26576
```

In case the smoothing variable in the model is a random variable, the `predict` method for the class attribute `"tvsure"` requires also a new set of values in argument `newz`. The chunk below first fits a TVSURE model, `tvOLS.z.fit`, to the `Kmenta` data with the same system of equations as in the `TVOLS.fit`, but with random variable as the smoothing variable, which is generated as an ARMA(2,2) process. Three values of the dependent variable are predicted with the `predict` method. In addition to new values in the argument `newdata`, it requires a set of new smoothing values in the argument `newz`. It returns the predicted values as a matrix with as many columns as equations in the system.

```
> nobs <- nrow (Kmenta)
> smoothing <- arima.sim(n = nobs + 3, sd = sqrt(0.1796),
+                        list(ar = c(0.8897, -0.4858), ma = c(-0.2279, 0.2488)))
> smoothing <- as.numeric(smoothing)
> tvOLS.z.fit <- tvSURE(system, data = Kmenta, z = smoothing[1:nobs])
> newSmoothing <- tail(smoothing, 3)
> predict(tvOLS.z.fit, newdata = newKmenta, newz = newSmoothing)

       demand    supply
[1,] 100.0195  96.50136
[2,] 100.3919 105.29293
[3,] 106.1426 107.97822
```

The `forecast` and `predict` methods for the rest of the class attributes in the package follow similar patterns, and further examples can be found in the documentation of the **tvReg**.

12 Summary

Research of time-varying coefficient linear models and their estimation using kernel smoothing methods has seen a great theoretical development during the last two decades. Our own work in this field has served as inspiration to code the **tvReg** because we encounter a lack of computer applications with this functionality. Indeed, we expect that this package empowers empirical researchers working with regression and time series models with a computing tool that allows for more flexible models.

Within the R framework: (i) the **tvReg** extends functions in the R packages **systemfit**, **plm** and **vars**; (ii) it extends functions `lm`, `ar.ols` and `arima` to allow for varying coefficients; (iv) it complements R packages **mgcv** and **gam** for the linear regression model by providing a consistent estimator of this model for in case of dependency and a general distribution in the error term; (v) it complements R package **mgm** by adding the time-varying impulse response (TVIRF) function which is commonly used in macroeconomics; and (vi) it complements R package **bvarsv** and **MARSS** which estimate the TVVAR and TVIRF within the state-space framework. In addition, the `confint`, `fitted`, `forecast`, `plot`, `predict`, `print`, `resid` and `summary` methods are implemented for all class attributes in the **tvReg** and will allow the user to conveniently produce their research output. In any case, the user is able to produce customised plots and summaries from the returns of the functions, whose elements are accessible in the same manner as other R `"list"` objects.

Finally, the **tvReg** shows multiple applications in economics and finance. Specifically in asset management, portfolio management, risk management, health policy and monetary policy. The methods and datasets permit to verify results in Aslanidis and Casas (2013); Casas et al. (2018, 2019, 2021). Models in this paper are used in other fields too. For example, Reikard (2009) uses the TVLM to forecast the wave energy flux and Haslbeck et al. (2021) uses the TVVAR in different applications in psychology. The **tvReg** is therefore not only relevant and original but also timely.

Acknowledgements

We thank the unknown referee for the constructive suggestions and comments on an earlier version of this paper. The first author would also like to thank her co-authors of related papers: Nektarios Aslanidis, Eva Ferreira, Jiti Gao, Stefano Grassi, Xiuping Mao, Susan Orbe, Bin Peng, Helena Veiga and Shangyu Xie whose comments, suggestions and collaboration have helped to shape this code. The first author acknowledges financial support from the European Research Council under the European Union's Horizon 2020 research and innovation program (grant no. 657182) and from the Ministerio de Economía y Competitividad (grant no. ECO2014-51914-P). The second author acknowledges financial support from the MINECO (grant no. MTM2017-82724-R), MICINN (grant no. PID2020-113578RB-I00), and from the Xunta de Galicia (Grupos de Referencia Competitiva ED431C-2020-14 and Centro de Investigación del Sistema Universitario de Galicia ED431G 2019/01), all of them through the ERDF.

Bibliography

T. G. Andersen and T. Bollerslev. Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International Economic Review*, 39:885–905, 1998. URL https://doi.org/10.2307/2527343. [p94]

N. Aslanidis and I. Casas. Nonparametric correlation models for portfolio allocation. *Journal of Banking & Finance*, 37:2268–2283, 2013. URL https://doi.org/10.1016/j.jbankfin.2013.01.010. [p91, 97]

T. Bollerslev, G. Tauchen, and H. Zhou. Expected stock returns and variance risk premia. *The Review of Financial Studies*, 22(11):44–63, 2009. URL https://doi.org/10.1093/rfs/hhp008. [p94]

T. Bollerslev, A. J. Patton, and R. Quaedvlieg. Exploiting the errors: A simple approach for improved volatility forecasting. *Journal of Econometrics*, 192(1):1–18, 2016. URL https://doi.org/10.1016/j.jeconom.2015.10.007. [p95]

Z. Cai. Trending time-varying coefficient time series with serially correlated errors. *Journal of Econometrics*, 136:163–188, 2007. URL https://doi.org/10.1016/j.jeconom.2005.08.004. [p92]

Z. Cai, J. Fan, and Q. Yao. Functional-coefficient regression models for nonlinear time series. *Journal of the American Statistical Association*, 95:941–956, 2000. URL https://doi.org/10.1080/01621459.2000.10474284. [p93]

Z. Cai, Q. Li, and J. Y. Park. Functional-coefficient models for nonstationary time series data. *Journal of Econometrics*, 148:101–113, 2009. URL https://doi.org/10.1016/j.jeconom.2008.10.003. [p93]

C. K. Carter and R. Kohn. On gibbs sampling for state space models. *Biometrika*, 81:541–553, 1994. URL https://doi.org/10.1093/biomet/81.3.541. [p80, 88]

I. Casas, X. Mao, and H. Veiga. Reexamining financial and economic predictability with new estimators of realized variance and variance risk premium. Working papers, CREATES, Aarhus University, 2018. URL http://econ.au.dk/fileadmin/site_files/filer_oekonomi/Working_Papers/CREATES/2018/rp18_10.pdf. [p95, 97]

I. Casas, E. Ferreira, and S. Orbe. Time-varying coefficient estimation in SURE models. application to portfolio management. *Journal of Financial Econometrics*, 19:707–745, 2019. URL https://doi.org/10.1093/jjfinec/nbz010. [p81, 85, 97]

I. Casas, J. Gao, B. Peng, and S. Xie. Time-varying income elasticities of healthcare expenditure for the OECD and Eurozone. *Journal of Applied Econometrics*, 36:328–345, 2021. URL https://doi.org/10.1002/jae.2809. [p82, 86, 97]

Y. Chang and E. Martinez-Chombo. Electricity demand analysis using cointegration and error-correction models with time varying parameters: The mexican case. Working papers, Rice University, Department of Economics, 2003. URL https://ideas.repec.org/p/ecl/riceco/2003-08.html. [p92, 93]

X. B. Chen, J. Gao, D. Li, and P. Silvapulle. Nonparametric estimation and forecasting for time-varying coefficient realized volatility models. *Journal of Business & Economic Statistics*, online:1–13, 2017. URL https://doi.org/10.1080/07350015.2016.1138118. [p86, 92, 94]

C. K. Chu and J. S. Marron. Comparison of two bandwidth selectors with dependent errors. *Annals of Statistics*, 19:1906–1918, 1991. URL https://doi.org/10.1214/aos/1176348377. [p83]

T. Cogley and T. J. Sargent. Drifts and volatilities: monetary policies and outcomes in the post wwii us. *Review of Economic Dynamics*, 8:262–302, 2005. URL https://doi.org/10.1016/j.red.2004.10.009. [p83]

F. Corsi. A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2):174–196, nov 2009. URL https://doi.org/10.1093/jjfinec/nbp001. [p92, 94]

Y. Croissant and G. Millo. Panel data econometrics in R: The plm package. *Journal of Statistical Software*, 27(2):1–43, 2008. URL https://doi.org/10.18637/jss.v027.i02. [p79]

Y. Croissant and G. Millo. *Panel Data Econometrics with R: the plm package*. Wiley, 2018. [p79]

R. Dahlhaus. Fitting time series models to nonstationary processes. *Annals of Statistics*, 25:1–37, 1997. URL https://doi.org/10.1214/aos/1034276620. [p83]

M. Das. Instrumental variables estimators of nonparametric models with discrete endogenous regressors. *Journal of Econometrics*, 124:335–361, 2005. URL https://doi.org/10.1016/j.jeconom.2004.02.001. [p93]

C. Dong, J. Gao, and B. Peng. Varying-coefficient panel data models with nonstationarity and partially observed factor structure. *Journal of Business & Economic Statistics*, 39:700–711, 2021. URL https://doi.org/10.1080/07350015.2020.1721294. [p82]

Dong, C. Jiti Gao, J. and Peng, B. Semiparametric single-index panel data models with cross-sectional dependence. *Journal of Econometrics*, 188:301–312, 2015. URL https://doi.org/10.1016/j.jeconom.2015.06.001. [p82]

R. L. Eubank. *Nonparametric Regression and Spline Smoothing*. New York: Marcel Dekker, 2nd edition, 1999. [p79]

E. Fama and K. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33:3–56, 1993. URL https://doi.org/10.1016/0304-405x(93)90023-5. [p85]

E. F. Fama and K. R. French. A five-factor asset pricing model. *Journal of Financial Economics*, 116(1):1–22, 2015. URL https://doi.org/10.1016/j.jfineco.2014.10.010. [p85]

E. F. Fama and K. R. French. International tests of a five-factor asset pricing model. *Journal of Financial Economics*, 123:441–463, 2017. URL https://doi.org/10.1016/j.jfineco.2016.11.004. [p85]

J. Fan and I. Gijbels. *Local Polynomial Modeling and Its Applications*. hapman and Hall, London, 1996. [p81]

J. Fan and W. Zhang. Simultaneous confidence bands and hypothesis testing in varying-coefficient models. *Scandinavian Journal of Statistics*, 27:715–731, 2000. URL https://doi.org/10.1111/1467-9469.00218. [p86]

K. R. French. Data library, 2016. URL http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html. Accessed: 2016-09-01. [p85]

J. Gao and P. Phillips. Functional coefficient nonstationary regression. Cowles Foundation Discussion Papers 1911, Cowles Foundation for Research in Economics, Yale University, 2013. URL https://ideas.repec.org/p/cwl/cwldpp/1911.html. [p93]

U.-G. Gerdtham, J. Soegaard, F. Andersson, and B. Jönsson. An econometric analysis of health care expenditure: A cross-section study of the OECD countries. *Journal of Health Economics*, 11:63–84, 1992. URL https://doi.org/10.1016/0167-6296(92)90025-v. [p86]

J. Haslbeck, L. Bringmann, and L. Waldorp. A tutorial on estimating time-varying vector autoregressive models. *Multivariate Behavioral Research*, 56:120–149, 2021. URL https://doi.org/10.1080/00273171.2020.1743630. [p97]

J. M. Haslbeck and L. J. Waldorp. mgm: Estimating time-varying mixed graphical models in high-dimensional data. *Journal of Statistical Software*, 93:1–46, 2020. URL https://doi.org/10.18637/jss.v093.i08. [p79]

T. Hastie. *gam: Generalized Additive Models*, 2022. URL https://CRAN.R-project.org/package=gam. R package version 1.20.1. [p80]

T. Hastie and R. Tibshirani. Varying-coefficient models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55:757–796, 1993. URL https://doi.org/10.1111/j.2517-6161.1993.tb01939.x. [p79, 93]

D. J. Henderson, S. C. Kumbhakar, Q. Li, and C. F. Parmeter. Smooth coefficient estimation of a seemingly unrelated regression. *Journal of Econometrics*, 189:148–162, 2015. URL https://doi.org/10.1016/j.jeconom.2015.07.002. [p81]

A. Henningsen and J. D. Hamann. systemfit: A package for estimating systems of simultaneous equations in R. *Journal of Statistical Software*, 23(4):1–40, 2007. URL https://doi.org/10.18637/jss.v023.i04. [p79]

E. E. Holmes, E. J. Ward, and K. Wills. Marss: Multivariate autoregressive state-space models for analyzing time-series data. *The R Journal*, 4(1):30, 2012. URL https://doi.org/10.32614/rj-2012-002. [p79]

G. Kapetanios, H. Mumtaz, I. Stevens, and K. Theodoridis. Assessing the economy-wide effects of quantitative easing. *The Economic Journal*, 122:316–347, 2012. URL http://www.jstor.org/stable/23324226. [p83]

G. Kapetanios, M. Marcellino, and F. Venditti. Large time-varying parameter VARs: a non-parametric approach. Temi di discussione (economic working papers), Bank of Italy, Economic Research and International Relations Area, 2017. URL https://ideas.repec.org/p/bdi/wptemi/td_1122_17.html. [p83]

J. Kmenta. *Elements of Econometrics*. New York: Macmillan, 2nd edition, 1986. [p84]

F. Krueger. *bvarsv: Bayesian analysis of a vector autoregressive model with stochastic volatility and time-varying parameters*, 2015. URL https://CRAN.R-project.org/package=bvarsv. R package version 1.1. [p80]

Y. Liu and F. Guo. A bayesian time-varying coefficient model for multitype recurrent events. *Journal of Computational and Graphical Statistics*, 29:383–395, 2020. URL https://doi.org/10.1080/10618600.2019.1686988. [p80]

H. Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer, 2005. [p83]

E. Mammen. Bootstrap and wild bootstrap for high dimensional linear models. *The Annals of Statistics*, 21:255–285, 1993. URL https://doi.org/10.1214/aos/1176349025. [p86]

S. Orbe, E. Ferreira, and J. Rodriguez-Poo. Nonparametric estimation of time varying parameters under shape restrictions. *Journal of Econometrics*, 126:53–77, 2005. URL https://doi.org/10.1016/j.jeconom.2004.02.006. [p81]

B. Pfaff. VAR, SVAR and SVEC models: Implementation within R package vars. *Journal of Statistical Software, Articles*, 27(4):1–32, 2008. URL https://www.jstatsoft.org/v027/i04. [p79]

G. E. Primiceri. Time varying structural vector autoregressions and monetary policy. *Review of Economic Studies*, 72:821–852, 2005. URL https://doi.org/10.1111/j.1467-937x.2005.00353.x. [p80, 83, 88, 89]

G. Reikard. Forecasting ocean wave energy: Tests of time-series models. *Ocean Engineering*, 36:348–356, 2009. URL https://doi.org/10.1016/j.oceaneng.2009.01.003. [p97]

P. Robinson. Nonparametric estimation of time-varying parameters. In P. Hackl, editor, *Statistical Analysis and Forecasting of Economic Structural Change*, pages 253–264. Springer, Berlin, 1989. URL https://doi.org/10.1007/978-3-662-02571-0_15. [p79, 92]

C. A. Sims. Macroeconomics and reality. *Econometrica*, 48:1–48, 1980. URL https://doi.org/10.2307/1912017. [p82]

H. S. M. D. Store. Intraday products, 2017. URL http://download-stock-data.webs.com/. Accessed: 2017-11-01. [p94]

Y. Sun, R. Carroll, and D. Li. Semiparametric estimation of fixed-effects panel data varying coefficient models. *Advances in Econometrics*, 25:101–129, 12 2009. URL https://doi.org/10.1108/s0731-9053(2009)0000025006. [p82]

Y. Sun, Z. Cai, and Q. Li. Semiparametric functional coefficient models with integrated covariates. *Econometric Theory*, 29:659–672, 2013. URL https://doi.org/10.1017/s0266466612000710. [p93]

S. N. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 2 edition, 2017. [p80]

Z. Xiao. Functional-coefficient cointegration models. *Journal of Econometrics*, 152:81–92, 2009. URL https://doi.org/10.1016/j.jeconom.2009.01.008. [p93]

Y. Yan, J. Gao, and B. Peng. Nonparametric time-varying vector moving average (infinity) models. Technical report, SSRN Electronic Journal, 2021. URL http://dx.doi.org/10.2139/ssrn.3729872. [p83]

A. Zellner. An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. *Journal of the American Statistical Association*, 57:348–368, 1962. URL https://doi.org/10.1080/01621459.1962.10480664. [p80]

T. Zhang and W. B. Wu. Inference of time-varying regression models. *Annals of Statistics*, 40(3): 1376–1402, 2012. URL https://doi.org/10.1214/12-aos1010. [p93]

Isabel Casas
Department of Economics and Finance
University of Deusto
Spain
ORCID: 0000-0002-9063-9670
casasis@gmail.com

Rubén Fernández-Casal
Department of Mathematics
University of A Coruña
Spain
ORCID: 0000-0002-5785-3739
ruben.fcasal@udc.es

RKHSMetaMod: An R Package to Estimate the Hoeffding Decomposition of a Complex Model by Solving RKHS Ridge Group Sparse Optimization Problem

by Halaleh Kamari, Sylvie Huet and Marie-Luce Taupin

Abstract In this paper, we propose an R package, called **RKHSMetaMod**, that implements a procedure for estimating a meta-model of a complex model. The meta-model approximates the Hoeffding decomposition of the complex model and allows us to perform sensitivity analysis on it. It belongs to a reproducing kernel Hilbert space that is constructed as a direct sum of Hilbert spaces. The estimator of the meta-model is the solution of a penalized empirical least-squares minimization with the sum of the Hilbert norm and the empirical L^2-norm. This procedure, called RKHS ridge group sparse, allows both to select and estimate the terms in the Hoeffding decomposition, and therefore, to select and estimate the Sobol indices that are non-zero. The **RKHSMetaMod** package provides an interface from the R statistical computing environment to the C++ libraries **Eigen** and **GSL**. In order to speed up the execution time and optimize the storage memory, except for a function that is written in R, all of the functions of this package are written using the efficient C++ libraries through **RcppEigen** and **RcppGSL** packages. These functions are then interfaced in the R environment in order to propose a user-friendly package.

1 Introduction

Consider a phenomenon described by a model m depending on d input variables $X = (X_1, ..., X_d)$. This model m from $\mathbb{R}^d$ to $\mathbb{R}$ may be a known model that is calculable in all points of X, i.e. $Y = m(X)$, or it may be an unknown regression model defined as follows:

$$Y = m(X) + \varepsilon, \tag{1}$$

where the error ε is assumed to be centered with a finite variance, i.e. $E(\varepsilon) = 0$ and $\text{var}(\varepsilon) < \infty$. The components of X are independent with a known law $P_X = \prod_{a=1}^{d} P_{X_a}$ on $\mathcal{X}$, a subset of $\mathbb{R}^d$. The number d of components of X may be large. The model m may present high complexity as strong non-linearities and high order interaction effects, and it is assumed to be square-integrable, i.e. $m \in L^2(\mathcal{X}, P_X)$. Based on the data points $\{(X_i, Y_i)\}_{i=1}^{n}$, we estimate a meta-model that approximates the Hoeffding decomposition of m. This meta-model belongs to a reproducing kernel Hilbert space (RKHS), which is constructed as a direct sum of the Hilbert spaces leading to an additive decomposition including variables and interactions between them (Durrande et al., 2013). The estimator of the meta-model is calculated by minimizing an empirical least-squares criterion penalized by the sum of two penalty terms: the Hilbert norm and the empirical norm (Huet and Taupin, 2017). This procedure allows us to select the subsets of variables $X_1, ..., X_d$ that contribute to predict Y. The estimated meta-model is used to perform sensitivity analysis, and therefore, to determine the influence of each variable and groups of them on the output variable Y.

In the classical framework of the sensitivity analysis, the function m is calculable in all points of X, and one may use the method of Sobol (1993) to perform the sensitivity analysis on m. Let us briefly introduce this method:
The independency between the components of X leads to write the function m according to its Hoeffding decomposition (Sobol, 1993; Van der Vaart, 1998):

$$m(X) = m_0 + \sum_{a=1}^{d} m_a(X_a) + \sum_{a<a'} m_{a,a'}(X_a, X_{a'}) + ... + m_{1,...,d}(X). \tag{2}$$

The terms in this decomposition are defined using the conditional expected values:

$$m_0 = E_X(m(X)),\ m_a(X_a) = E_X(m(X)|X_a) - m_0;$$
$$m_{a,a'}(X_a, X_{a'}) = E_X(m(X)|X_a, X_{a'}) - m_a(X_a) - m_{a'}(X_{a'}) - m_0, \cdots$$

These terms are known as the constant term, main effects, interactions of order two and so on. Let $\mathcal{P}$ be the set of all subsets of $\{1, ..., d\}$ with dimension 1 to d. For all $v \in \mathcal{P}$ and $X \in \mathcal{X}$, let X_v be the vector

with components X_a, $a \in v$. For a set A let $|A|$ be its cardinality, and for all $v \in \mathcal{P}$, let $m_v : \mathbb{R}^{|v|} \to \mathbb{R}$ be the function associated with X_v in Equation (2). Then Equation (2) can be expressed as follows:

$$m(X) = m_0 + \sum_{v \in \mathcal{P}} m_v(X_v). \tag{3}$$

This decomposition is unique, all terms m_v, $v \in \mathcal{P}$ are centered, and they are orthogonal with respect to $L^2(\mathcal{X}, P_X)$. The functions m and m_v, $v \in \mathcal{P}$ in Equation (3) are square-integrable. As any two terms of decomposition (3) are orthogonal, by squaring (3) and integrating it with respect to the distribution of X, a decomposition of the variance of $m(X)$ is obtained as follows:

$$\text{var}(m(X)) = \sum_{v \in \mathcal{P}} \text{var}(m_v(X_v)). \tag{4}$$

The Sobol indices associated with the group of variables X_v, $v \in \mathcal{P}$ are defined by:

$$S_v = \text{var}(m_v(X_v))/\text{var}(m(X)). \tag{5}$$

For each v, the S_v expresses the fraction of the variance of $m(X)$ explained by X_v. For all $v \in \mathcal{P}$, when $|v| = 1$, the S_vs are referred to as the first order indices; when $|v| = 2$, i.e. $v = \{a, a'\}$ and $a \neq a'$, they are referred to as the second order indices or the interaction indices of order two (between X_a and $X_{a'}$); and the same holds for $|v| > 2$.

The total number of the Sobol indices to be calculated is equal to $|\mathcal{P}| = 2^d - 1$, which raises exponentially with the number of the input variables d. When d is large, the evaluation of all the indices can be computationally demanding and even not reachable. In practice, only the indices of order not higher than two are calculated. However, only the first and second order indices may not provide a good information on the model sensitivities. In order to provide better information on the model sensitivities, Homma and Saltelli (1996) proposed to calculate the first order and the total indices defined as follows:
Let $\mathcal{P}_a \subset \mathcal{P}$ be the set of all the subsets of $\{1, ..., d\}$ including a, then $S_{T_a} = \sum_{v \in \mathcal{P}_a} S_v$. For all $a \in \{1, ..., d\}$, the S_{T_a} denotes the total effect of X_a. It expresses the fraction of variance of $m(X)$ explained by X_a alone and all the interactions of it with the other variables. The total indices allow us to rank the input variables with respect to the amount of their effect on the output variable. However, they do not provide complete information on the model sensitivities as do all the Sobol indices.

The classical computation of the Sobol indices is based on the Monte Carlo methods (see for example: Sobol (1993) for the main effect and interaction indices, and Saltelli (2002) for the main effect and total indices). For models that are expensive to evaluate, the Monte Carlo methods lead to a high computational burden. Moreover, in the case where d is large, m is complex and the calculation of the variances (see Equation (4)) is numerically complicated or not possible (as in the case where the model m is unknown) the methods described above are not applicable. Another approach consists in approximating m by a simplified model, called a meta-model, which is much faster to evaluate and to perform sensitivity analysis on it. Beside the approximations of the Sobol indices of m at a lower computational cost, a meta-model provides a deeper view of the input variables effects on the model output. Among the meta-modelling methods proposed in the literature, the expansion based on the polynomial Chaos (Wiener, 1938; Schoutens, 2000) can be used to approximate the Hoeffding decomposition of m (Sudret, 2008). The principle of the polynomial Chaos is to project m onto a basis of orthonormal polynomials. The polynomial Chaos expansion of m is written as (Soize and Ghanem, 2004):

$$m(X) = \sum_{j=0}^{\infty} h_j \phi_j(X), \tag{6}$$

where $\{h_j\}_{j=0}^{\infty}$ are the coefficients, and $\{\phi_j\}_{j=0}^{\infty}$ are the multivariate orthonormal polynomials associated with X which are determined according to the distribution of the components of X. In practice, expansion (6) shall be truncated for computational purposes, and the model m may be approximated by $\sum_{j=0}^{v_{max}} h_j \phi_j(X)$, where v_{max} is determined using a *truncation scheme*. The Sobol indices are obtained then by summing up the squares of the suitable coefficients. Blatman and Sudret (2011) proposed a method for truncating the polynomial Chaos expansion and an algorithm based on the least angle regression for selecting the terms in the expansion. In this approach, according to the distribution of the components of X, a unique family of orthonormal polynomials $\{\phi_j\}_{j=0}^{\infty}$ is determined. However, this family may not be necessarily the best functional basis to approximate m well.

Gaussian Process (GP) can also be used to construct meta-models as highlighted in Welch et al. (1992), Oakley and O'Hagan (2004), Kleijnen (2007, 2009), Marrel et al. (2009), Durrande et al. (2012), and Le Gratiet et al. (2014). The principle is to consider that the prior knowledge about the function $m(X)$, can be modelled by a GP $\mathcal{Z}(X)$ with a mean $m_{\mathcal{Z}}(X)$ and a covariance kernel $k_{\mathcal{Z}}(X, X')$. To

perform sensitivity analysis from a GP model, one may replace the model $m(X)$ with the mean of the conditional GP and deduce the Sobol indices from it. A review on the meta-modelling based on the polynomial Chaos and the GP is presented in Le Gratiet et al. (2017).

Durrande et al. (2013) considered a class of the functional approximation methods similar to the GP and obtained a meta-model that satisfies the properties of the Hoeffding decomposition. They proposed to approximate m by functions belonging to a RKHS $\mathcal{H}$ which is a direct sum of the Hilbert spaces. Their RKHS $\mathcal{H}$ is constructed in a way that the projection of m onto $\mathcal{H}$, denoted f^*, is an approximation of the Hoeffding decomposition of m. The function f^* is defined as the minimizer over the functions $f \in \mathcal{H}$ of the criterion $E_X(m(X) - f(X))^2$.

Let $\langle .,. \rangle \mathcal{H}$ be the scalar product in $\mathcal{H}$, let also k and k_v be the reproducing kernels associated with the RKHS $\mathcal{H}$ and the RKHS $\mathcal{H}_v$ respectively. The properties of the RKHS $\mathcal{H}$ insures that any function $f \in \mathcal{H}$, $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ is written as the following decomposition:

$$f(X) = \langle f, k(X,.) \rangle \mathcal{H} = f_0 + \sum_{v \in \mathcal{P}} f_v(X_v), \tag{7}$$

where f_0 is constant, and $f_v : \mathbb{R}^{|v|} \to \mathbb{R}$ is defined by $f_v(X) = \langle f, k_v(X,.) \rangle_{\mathcal{H}}$. For more details on the RKHS construction and the definition of the Hilbert norm see Section "RKHS construction" in the Appendix (supplementary materials).

For all $v \in \mathcal{P}$, the functions $f_v(X_v)$ are centered and for all $v \neq v'$, the functions $f_v(X_v)$ and $f_{v'}(X_{v'})$ are orthogonal with respect to $L^2(\mathcal{X}, P_X)$. Therefore, the decomposition of the function f presented in Equation (7) is its Hoeffding decomposition. As the function f^* belongs to the RKHS $\mathcal{H}$, it is decomposed as its Hoeffding decomposition, $f^* = f_0^* + \sum_{v \in \mathcal{P}} f_v^*$, and each function f_v^* approximates the function m_v in Equation (3). The number of the terms f_v^* that should be estimated in the Hoeffding decomposition of f^* is equal to $|\mathcal{P}| = 2^d - 1$, which may be huge since it rises very quickly by increasing d. In order to deal with this problem, in the regression framework, one may estimate f^* by a sparse meta-model $\widehat{f} \in \mathcal{H}$. To this end, the estimation of f^* is done on the basis of n observations by minimizing a least-squares criterion suitably penalized in order to deal with both the non-parametric nature of the problem and the possibly large number of functions that have to be estimated. In the classical framework of the sensitivity analysis one may calculate a sparse approximation of f^* using least-squares penalized criterion as it is done in the non-parametric regression framework. In order to obtain a sparse solution of a minimization problem, the penalty function should enforce the sparsity. There exists various ways of enforcing sparsity for a minimization (maximization) problem, see for example Hastie et al. (2015) for a review. Some methods, such as the Sparse Additive Models (SpAM) procedure (Ravikumar et al., 2009; Liu et al., 2009) are based on a combination of the l_1-norm with the empirical L^2-norm: $\|f\|_{n,1} = \sum_{a=1}^d \|f_a\|_n$, where $\|f_a\|_n^2 = \sum_{i=1}^n f_a^2(X_{ai})/n$, is the squared empirical L^2-norm of the univariate function f_a. The Component Selection and Smoothing Operator (COSSO) method developed by Lin and Zhang (2006) enforces sparsity using a combination of the l_1-norm with the Hilbert norm: $\|f\|_{\mathcal{H},1} = \sum_{a=1}^d \|f_a\|_{\mathcal{H}_a}$. Instead of focusing on only one penalty term, one may consider a more general family of estimators, called the *doubly penalized estimator*, which is obtained by minimizing a criterion penalized by the sum of two penalty terms. Raskutti et al. (2009, 2012) proposed a *doubly penalized estimator*, which is the solution of the minimization of a least-squares criterion penalized by the sum of a sparsity penalty term and a combination of the l_1-norm with the Hilbert norm:

$$\gamma \|f\|_{n,1} + \mu \|f\|_{\mathcal{H},1}, \tag{8}$$

where $\gamma, \mu \in \mathbb{R}$ are the tuning parameters that should be suitably chosen.

Meier et al. (2009) proposed a related family of estimators, based on the penalization with the empirical L^2-norm. Their penalty function is the sum of the sparsity penalty term, $\|f\|_{n,1}$, and a smoothness penalty term. Huet and Taupin (2017) considered the same approximation functional spaces as Durrande et al. (2013), and obtained a *doubly penalized estimator* of a meta-model which approximates the Hoeffding decomposition of m. Their estimator is the solution of the least-squares minimization penalized by the penalty function defined in Equation (8) adapted to the multivariate setting,

$$\gamma \|f\|_n + \mu \|f\|_{\mathcal{H}}, \text{ with } \|f\|_n = \sum_{v \in \mathcal{P}} \|f_v\|_n, \ \|f\|_{\mathcal{H}} = \sum_{v \in \mathcal{P}} \|f_v\|_{\mathcal{H}_v}. \tag{9}$$

This procedure, called RKHS ridge group sparse, estimates the groups v that are suitable for predicting f^*, and the relationship between f_v^* and X_v for each group. The obtained estimator, called RKHS meta-model, is used then to estimate the Sobol indices of m. This approach renders it possible to estimate the Sobol indices for all groups in the support of the RKHS meta-model, including the interactions of possibly high order, a point known to be difficult in practice.

In this paper, we introduce an R package, called **RKHSMetaMod**, that implements the RKHS ridge group sparse procedure. The functions of this package allows us to:

(1) calculate the reproducing kernels and their associated Gram matrices (see Section Calculation of the Gram matrices),

(2) implement the RKHS ridge group sparse procedure and a special case of it, called the RKHS group lasso procedure (when $\gamma = 0$ in the penalty function (9)), in order to estimate the terms f_v^* in the Hoeffding decomposition of the meta-model f^* leading to an estimation of the function m (see Section Optimization algorithms),

(3) choose the tuning parameters μ and γ (see Equation (9)), using a procedure that leads to obtain the *best* RKHS meta-model in terms of the prediction quality,

(4) estimate the Sobol indices of the function m (see Section Estimation of the Sobol indices).

The current version of the package supports uniformly distributed input variables on $\mathcal{X} = [0,1]^d$. However, it could be easily adapted to datasets with input variables from another distribution by making a small modification to one of its functions (see Remark 3 of Section Calculation of the Gram matrices).

Let us give a brief overview of the related existing statistical packages to the **RKHSMetaMod** package. The R package **sensitivity** is designed to implement sensitivity analysis methods and provides the approaches for numerical calculation of the Sobol indices. In particular, Kriging method can be used to reduce the number of the observations in global sensitivity analysis. The function `sobolGP` of the package **sensitivity** builds a Kriging based meta-model using the function `km` of the package **DiceKriging** (Roustant et al., 2012), and estimates its Sobol indices. This procedure can also be done using the function `km` and the function `fast99` of the package **sensitivity** (see Section 4.5. of Roustant et al. (2012)). In this case, the idea is once again to build a Kriging based meta-model using the function `km` and then estimate its Sobol indices using the function `fast99`. In both cases the true function is substituted by a Kriging based meta-model and then its Sobol indices are estimated. In the `sobolGP` function, the Sobol indices are estimated by the Monte Carlo integration, while the `fast99` function estimates them using the extended-FAST method (Saltelli et al., 1999). To reduce the computational burden when dealing with large datasets and complex models, in **RKHSMetaMod** package, we propose to use the empirical variances to estimate the Sobol indices (see Section Estimation of the Sobol indices). Besides, the estimation of the Sobol indices in the **RKHSMetaMod** package is done based on the RKHS meta-model which is a sparse estimator. It is beneficial since instead of calculating the Sobol indices of all groups $v \in \mathcal{P}$, only the Sobol indices associated with the groups in the support of the RKHS meta-model are computed (see Section Estimation of the Sobol indices). Moreover, the functions `sobolGP` and `fast99` provide the estimation of the first order and the total Sobol indices only, while the procedure in the **RKHSMetaMod** package makes it possible to estimate the high order Sobol indices. The R packages **DiceKriging** and **DiceOptim** (Deep Inside Computer Experiments Kriging/Optim) (Roustant et al., 2012) implement the Kriging based meta-models to estimate complex models in the high dimensional context. They provide different GP (Kriging) models corresponding to the Gaussian, Matérn, Exponential and Power-Exponential correlation functions. The estimation of the parameters of the correlation functions in these packages relies on the global optimizer with gradient `genoud` algorithm of the package **rgenoud** (Mebane and Sekhon, 2011). These packages do not implement any method of the sensitivity analysis themselves. However, some authors (see Section 4.5. of Roustant et al. (2012) for example) perform sensitivity analysis on their estimated meta-models by employing the functions of the package **sensitivity**. The R package **RobustGaSP** (Robust Gaussian Stochastic Process) (Gu et al., 2019) approximates a complex model by a GP meta-model. This package implements marginal posterior mode estimation of the GP model parameters. The estimation method in this package insures the robustness of the parameter estimation in the GP model, and allows one also to identify input variables that have no effect on the variability of the function under study. The R package **mlegp** (maximum likelihood estimates of Gaussian processes) (Dancik and Dorman, 2008) provides functions to implement both meta-modelling approaches and sensitivity analysis methods. It obtains maximum likelihood estimates of the GP model for the output of costly computer experiments. The GP models are built either on the basis of the Gaussian correlation function or on the basis of the first degree polynomial trend. The sensitivity analysis methods implemented in this package include Functional Analysis of Variance (FANOVA) decomposition, plot functions to obtain diagnostic plots, main effects, and second order interactions. The prediction quality of the meta-model depends on the quality of the estimation of its parameters and more precisely the estimation of parameters in the correlation functions (Kennedy and O'Hagan, 2000). The maximum likelihood estimation of these parameters often produce unstable results, and as a consequence, the obtained meta-model may have an inferior prediction quality (Gu et al., 2018; Gu, 2019). The **RKHSMetaMod** package is devoted to the meta-model estimation on the RKHS $\mathcal{H}$. It implements the convex optimization algorithms to calculate meta-models; provides the functions to compute the prediction error of the obtained meta-models; performs the sensitivity analysis on the obtained meta-models and more precisely calculate their Sobol

indices. The convex optimization algorithms used in this package are all written using C++ libraries, and are adapted to take into account the problem of high dimensionality in this context. This package is available from the Comprehensive R Archive Network (CRAN) (Kamari, 2019).

The organization of the paper is as follows: In the next Section, we describe the estimation method. In Section Algorithms, we present in details the algorithms used in the **RKHSMetaMod** package. Section RKHSMetaMod through examples includes two parts: In the first part, Section Simulation study, the performance of the **RKHSMetaMod** package functions is validated through a simulation study. In the second part, Section Comparison examples, the comparison in terms of the predictive accuracy between the RKHS meta-model and the Kriging based meta-models from **RobustGaSP** (Gu et al., 2019) and **DiceKriging** (Roustant et al., 2012) packages is given through two examples.

2 Estimation method

In this Section, we present: the RKHS ridge group sparse and the RKHS group lasso procedures (see RKHS ridge group sparse and RKHS group lasso procedures), the strategy of choosing the tuning parameters in the RKHS ridge group sparse algorithm (see Choice of the tuning parameters), and the calculation of the empirical Sobol indices of the RKHS meta-model (see Estimation of the Sobol indices).

RKHS ridge group sparse and RKHS group lasso procedures

Let us denote by n the number of observations. The dataset consists of a vector of n observations $Y = (Y_1, ..., Y_n)$, and a $n \times d$ matrix of features X with components $(X_{ai}, i = 1, ..., n, a = 1, ..., d) \in \mathbb{R}^{n\times d}$. For some tuning parameters γ_v, μ_v, $v \in \mathcal{P}$, the RKHS ridge group sparse criterion is defined by,

$$\mathcal{L}(f) = \frac{1}{n}\sum_{i=1}^{n}\Big(Y_i - f_0 - \sum_{v\in\mathcal{P}} f_v(X_{vi})\Big)^2 + \sum_{v\in\mathcal{P}}\gamma_v\|f_v\|_n + \sum_{v\in\mathcal{P}}\mu_v\|f_v\|_{\mathcal{H}_v}, \tag{10}$$

where X_v represents the matrix of variables corresponding to the v-th group, i.e. $X_v = (X_{vi}, i = 1, ..., n, v \in \mathcal{P}) \in \mathbb{R}^{n\times|\mathcal{P}|}$, and where $\|f_v\|_n$ is the empirical L^2-norm of f_v defined by the sample $\{X_{vi}\}_{i=1}^n$ as $\|f_v\|_n = \sqrt{\sum_{i=1}^n f_v^2(X_{vi})/n}$.

The penalty function in the criterion (10) is the sum of the Hilbert norm and the empirical norm, which allows us to select few terms in the additive decomposition of f over sets $v \in \mathcal{P}$. Moreover, the Hilbert norm favours the smoothness of the estimated f_v, $v \in \mathcal{P}$.
Let $\mathcal{F} = \{f : f = f_0 + \sum_{v\in\mathcal{P}} f_v,\ f_v \in \mathcal{H}_v,\ \|f_v\|_{\mathcal{H}_v} \le r_v,\ r_v \in \mathbb{R}^+\}$ be the set of functions. Then the RKHS meta-model is defined by,

$$\widehat{f} = \arg\min_{f\in\mathcal{F}} \mathcal{L}(f). \tag{11}$$

According to the Representer Theorem (Kimeldorf and Wahba, 1970), the non-parametric functional minimization problem described above is equivalent to a parametric minimization problem. Indeed, the solution of the minimization problem (11) belonging to the RKHS $\mathcal{H}$ is written as $f = f_0 + \sum_{v\in\mathcal{P}} f_v$, where for some matrix $\theta = (\theta_{vi}, i = 1, ..., n, v \in \mathcal{P}) \in \mathbb{R}^{n\times|\mathcal{P}|}$ we have for all $v \in \mathcal{P}$,

$$f_v(.) = \sum_{i=1}^{n}\theta_{vi}k_v(X_{vi}, .), \text{ and } \|f_v\|^2_{\mathcal{H}_v} = \sum_{i,i'=1}^{n}\theta_{vi}\theta_{vi'}k_v(X_{vi}, X_{vi'}). \tag{12}$$

Let $\|.\|$ be the Euclidean norm (called also L^2-norm) in $\mathbb{R}^n$, and for each $v \in \mathcal{P}$, let K_v be the $n \times n$ Gram matrix associated with the kernel $k_v(.,.)$, i.e. $(K_v)_{i,i'} = k_v(X_{vi}, X_{vi'})$. Let also $K^{1/2}$ be the matrix that satisfies $t(K^{1/2})K^{1/2} = K$, and let $\widehat{f}_0$ and $\widehat{\theta}$ be the minimizers of the following penalized least-squares criterion:

$$C(f_0, \theta) = \|Y - f_0 I_n - \sum_{v\in\mathcal{P}} K_v\theta_v\|^2 + \sqrt{n}\sum_{v\in\mathcal{P}}\gamma_v\|K_v\theta_v\| + n\sum_{v\in\mathcal{P}}\mu_v\|K_v^{1/2}\theta_v\|.$$

Then the estimator $\widehat{f}$ defined in Equation (11) satisfies,

$$\widehat{f}(X) = \widehat{f}_0 + \sum_{v\in\mathcal{P}}\widehat{f}_v(X_v) \text{ with } \widehat{f}_v(X_v) = \sum_{i=1}^{n}\widehat{\theta}_{vi}k_v(X_{vi}, X_v).$$

Remark 1 *The constraint $\|f_v\|_{\mathcal{H}_v} \le r_v$ is crucial for theoretical properties, but the value of r_v is generally*

unknown and has no practical usefulness. In this package, it is not taken into account in the parametric minimization problem.

For each $v \in \mathcal{P}$, let γ'_v and μ'_v be the weights that are chosen suitably. We define $\gamma_v = \gamma \times \gamma'_v$ and $\mu_v = \mu \times \mu'_v$ with $\gamma, \mu \in \mathbb{R}^+$.

Remark 2 *This formulation simplifies the choice of the tuning parameters since instead of tuning $2 \times |\mathcal{P}|$ parameters γ_v and μ_v, $v \in \mathcal{P}$, only two parameters γ and μ are tuned. Moreover, the weights γ'_v and μ'_v, $v \in \mathcal{P}$ may be of interest in practice. For example, one can take weights that increase with the cardinal of v in order to favour the effects with small interaction order between variables.*

For the sake of simplicity, in the rest of this paper for all $v \in \mathcal{P}$ the weights γ'_v and μ'_v are assumed to be set as one, and the RKHS ridge group sparse criterion is then expressed as follows:

$$C(f_0, \theta) = \|Y - f_0 I_n - \sum_{v \in \mathcal{P}} K_v \theta_v\|^2 + \sqrt{n}\gamma \sum_{v \in \mathcal{P}} \|K_v \theta_v\| + n\mu \sum_{v \in \mathcal{P}} \|K_v^{1/2} \theta_v\|. \tag{13}$$

If we consider only the second part of the penalty function in the criterion (13) (i.e. set $\gamma = 0$), we obtain the RKHS group lasso criterion as follows:

$$C_g(f_0, \theta) = \|Y - f_0 I_n - \sum_{v \in \mathcal{P}} K_v \theta_v\|^2 + n\mu \sum_{v \in \mathcal{P}} \|K_v^{1/2} \theta_v\|, \tag{14}$$

which is a group lasso criterion (Yuan and Lin, 2006) up to a scale transformation.

In the **RKHSMetaMod** package, the RKHS ridge group sparse algorithm is initialized using the solutions obtained by solving the RKHS group lasso algorithm. Indeed, the penalty function in the RKHS group lasso criterion (14) insures the sparsity in the solution. Therefore, for a given value of μ, by implementing the RKHS group lasso algorithm (see Section RKHS group lasso), a RKHS meta-model with few terms in its additive decomposition is obtained. The support and the coefficients of a RKHS meta-model which is obtained by implementing the RKHS group lasso algorithm will be denoted by $\widehat{S}_{\widehat{f}_{\text{Group Lasso}}}$ and $\widehat{\theta}_{\text{Group Lasso}}$, respectively. From now on, we denote the tuning parameter in the RKHS group lasso criterion by:

$$\mu_g = \sqrt{n}\mu. \tag{15}$$

Choice of the tuning parameters

While dealing with an optimization problem of a criterion of the form (13), one of the essential steps is to choose the appropriate tuning parameters. Cross-validation is generally used for that purpose. Nevertheless in the context of high-dimensional complex models, the computational time for a cross-validation procedure may be prohibitively high. Therefore, we propose a procedure based on a single testing dataset:

- we first choose, a grid of values of the tuning parameters μ and γ;
  Let $\mu_{\max}$ be the smallest value of μ_g (see Equation (15)), such that the solution to the minimization of the RKHS group lasso problem for all $v \in \mathcal{P}$ is $\theta_v = 0$. We have,

  $$\mu_{\max} = \max_v \left(2\|K_v^{1/2}(Y - \overline{Y})\| \right) / \sqrt{n}. \tag{16}$$

  In order to set up the grid of values of μ, one may find $\mu_{\max}$ and then a grid of values of μ could be defined by $\mu_l = \mu_{\max} / (\sqrt{n} \times 2^l)$ for $l \in \{1, ..., l_{\max}\}$. The grid of values of γ is chosen by the user.
- next, for the grid of values of μ and γ, we calculate a sequence of estimators. Each estimator associated with the pair (μ, γ) in the grid of values of μ and γ, denoted by $\widehat{f}_{(\mu,\gamma)}$, is the solution of the RKHS ridge group sparse optimization problem or the RKHS group lasso optimization problem if $\gamma = 0$.
- finally, the obtained estimators $\widehat{f}_{(\mu,\gamma)}$ are evaluated using a testing dataset, $\{(Y_i^{\text{test}}, X_i^{\text{test}})\}_{i=1}^{n^{\text{test}}}$. The prediction error associated with each estimator $\widehat{f}_{(\mu,\gamma)}$ is calculated by,

  $$\text{ErrPred}(\mu, \gamma) = \sum_{i=1}^{n^{\text{test}}} (Y_i^{\text{test}} - \widehat{f}_{(\mu,\gamma)}(X_i^{\text{test}}))^2 / n^{\text{test}},$$

where for $S_{\widehat{f}}$ being the support of the estimator $\widehat{f}_{(\mu,\gamma)}$ we have,

$$\widehat{f}_{(\mu,\gamma)}(X^{\text{test}}) = \widehat{f}_0 + \sum_{v\in S_{\widehat{f}}} \sum_{i=1}^{n} \widehat{\theta}_{vi} k_v(X_{vi}, X_v^{\text{test}}).$$

The pair $(\widehat{\mu}, \widehat{\gamma})$ with the smallest value of the prediction error is chosen, and the estimator $\widehat{f}_{(\widehat{\mu},\widehat{\gamma})}$ is considered as the *best* estimator of the function m, in terms of the prediction error.

In the **RKHSMetaMod** package, the algorithm to calculate a sequence of the RKHS meta-models, the value of $\mu_{\max}$, and the prediction error are implemented as `RKHSMetMod`, `mu_max`, and `PredErr` functions, respectively. These functions are described in Section "Overview of the RKHSMetaMod functions" (supplementary materials), and illustrated in Example 1, Example 2, and Examples 1, 2, 3, respectively.

Estimation of the Sobol indices

The variance of the function m is estimated by the variance of the estimator $\widehat{f}$. As the estimator $\widehat{f}$ belongs to the RKHS $\mathcal{H}$, it admits the Hoeffding decomposition and,

$$\text{var}(\widehat{f}(X)) = \sum_{v\in\mathcal{P}} \text{var}(\widehat{f}_v(X_v)), \text{ where } \forall v \in \mathcal{P},\ \text{var}(\widehat{f}_v(X_v)) = E_X(\widehat{f}_v^2(X_v)) = \|\widehat{f}_v\|_2^2.$$

In order to reduce the computational cost in practice, one may estimate the variances of $\widehat{f}_v(X_v)$, $v \in \mathcal{P}$ by their empirical variances. Let $\widehat{f}_{v.}$ be the empirical mean of $\widehat{f}_v(X_{vi})$, $i = 1, ..., n$, then:

$$\widehat{\text{var}}(\widehat{f}_v(X_v)) = \frac{1}{n-1} \sum_{i=1}^{n} (\widehat{f}_v(X_{vi}) - \widehat{f}_{v.})^2.$$

For the groups v that do not belong to the support of $\widehat{f}$, we have $\widehat{S}_v = 0$ and for the groups v that belong to the support of $\widehat{f}$, the estimators of the Sobol indices of m are defined by,

$$\widehat{S}_v = \widehat{\text{var}}(\widehat{f}_v(X_v)) / \sum_{v\in\mathcal{P}} \widehat{\text{var}}(\widehat{f}_v(X_v)).$$

In the **RKHSMetaMod** package, the algorithm to calculate the empirical Sobol indices $\widehat{S}_v$, $v \in \mathcal{P}$ is implemented as `SI_emp` function. This function is described in Section "Companion functions" (supplementary materials) and illustrated in Examples 1, 2, 3.

3 Algorithms

The **RKHSMetaMod** package implements two optimization algorithms: the RKHS ridge group sparse (see Algorithm 2) and the RKHS group lasso (see Algorithm 1). These algorithms rely on the Gram matrices K_v, $v \in \mathcal{P}$ that have to be positive definite. Therefore, the first and essential step in this package is to calculate these matrices and insure their positive definiteness. The algorithm of this step is described in the next Section. The second step is to estimate the RKHS meta-model. In the **RKHSMetaMod** package, two different objectives based on different procedures are considered to calculate this estimator:

1. The RKHS meta-model with the *best* prediction quality.
   The procedure to calculate the RKHS meta-model with the *best* prediction quality has been described in Section Choice of the tuning parameters: a sequence of values of the tuning parameters (μ, γ) is considered, and the RKHS meta-models associated with each pair of the values of (μ, γ) are calculated. For $\gamma = 0$, the RKHS meta-model is obtained by solving the RKHS group lasso optimization problem, while for $\gamma \neq 0$ the RKHS ridge group sparse optimization problem is solved to calculate the RKHS meta-model. The obtained estimators are evaluated by considering a new dataset and the RKHS meta-model with the minimum value of the prediction error is chosen as the *best* estimator.
2. The RKHS meta-model with at most $qmax$ groups in its support, i.e. $|S_{\widehat{f}}| \leq qmax$.
   First, the tuning parameter γ is set as zero. Then, a value of μ for which the number of groups $v \in \mathcal{P}$ in the solution of the RKHS group lasso optimization problem is equal to $qmax$, is computed. This value of μ will be denoted by μ_{qmax}. Finally, the RKHS meta-models containing at most $qmax$ groups in their support are obtained by implementing the RKHS ridge group

sparse algorithm for a grid of values of $\gamma \neq 0$ and μ_{qmax}. This procedure is described in more details in Section RKHS meta-model with *qmax* active groups.

Calculation of the Gram matrices

The available kernels in the **RKHSMetaMod** package are: Gaussian kernel, Matérn 3/2 kernel, Brownian kernel, quadratic kernel and linear kernel. The usual presentation of these kernels is given in Table 1. The choice of kernel, that is done by the user, determines the functional approximation space.

Kernel type	Mathematical formula for $u \in \mathbb{R}^n, v \in \mathbb{R}$	RKHSMetaMod name
Gaussian	$k_a(u,v) = \exp(-\|u-v\|^2/2r^2)$	`"gaussian"`
Matérn 3/2	$k_a(u,v) = (1+\sqrt{3}\|u-v\|/r)\exp(-\sqrt{3}\|u-v\|/r)$	`"matern"`
Brownian	$k_a(u,v) = \min(u,v)+1$	`"brownian"`
Quadratic	$k_a(u,v) = (u^T v+1)^2$	`"quad"`
Linear	$k_a(u,v) = u^T v+1$	`"linear"`

Table 1: List of the reproducing kernels used to construct the RKHS $\mathcal{H}$. The range parameters r in the Gaussian and Matérn 3/2 kernels are assumed to be fixed and set as $1/2$ and $\sqrt{3}/2$, respectively. The value 1 is added to the Brownian kernel to relax the constraint of nullity at the origin (Durrande et al., 2013).

For a chosen kernel, the algorithm to calculate the Gram matrices K_v, $v \in \mathcal{P}$ in the **RKHSMetaMod** package, is implemented as `calc_Kv` function. This algorithm is based on three essential points:

(1) Modify the chosen kernel:
In order to satisfy the conditions of constructing the RKHS $\mathcal{H}$ described in Section "RKHS construction" of the Appendix (supplementary materials), these kernels are modified according to Equation "(2)" (see the Appendix (supplementary materials)). Let us take the example of the Brownian kernel:
The RKHS associated with the Brownian kernel $k_a(X_a, X_a') = \min(X_a, X_a') + 1$ is well known to be $\mathcal{H}_a = \{f : [0,1] \to \mathbb{R} \text{ is absolutely continuous, and } f(0) = 0,\ \int_0^1 f'(X_a)^2 dX_a < \infty\}$, with the inner product $\langle f, h\rangle_{\mathcal{H}_a} = \int_0^1 f'(X_a)h'(X_a)dX_a$. Easy calculations lead to obtain the Brownian kernel as follows,

$$k_{0a} = \min(X_a, X_a') + 1 - (3/4)(1 + X_a - X_a^2/2)(1 + X_a' - X_a'^2/2).$$

The RKHS associated with kernel k_{0a} is the set $\mathcal{H}_{0a} = \{f \in \mathcal{H}_a : \int_0^1 f(X_a)dX_a = 0\}$, and we have $\mathcal{H} = \mathbb{1} + \sum_{v\in\mathcal{P}} \mathcal{H}_v = \{f : [0,1]^d \to \mathbb{R} : f = f_0 + \sum_{v\in\mathcal{P}} f_v(X_v),\ \text{with } f_v \in \mathcal{H}_v\}$.

Remark 3 *In the current version of the package, we consider the input variables $X = (X_1, ..., X_d)$ that are uniformly distributed on $[0,1]^d$. In order to consider the input variables that are not distributed uniformly, it suffices to modify a part of the function* `calc_Kv` *related to the calculation of the kernels k_{0a}, $a = 1, ..., d$. For example, for $X = (X_1, ..., X_d)$ being distributed with law $P_X = \prod_{a=1}^d P_a$ on $\mathcal{X} = \bigotimes_{a=1}^d \mathcal{X}_a \subset \mathbb{R}^d$, the kernel k_{0a} associated with the Brownian kernel is calculated as follows,*

$$k_{0a} = \min(X_a, X_a') + 1 - \frac{(\int_{\mathcal{X}_a}(\min(X_a, U)+1)dP_a)(\int_{\mathcal{X}_a}(\min(X_a', U)+1)dP_a)}{(\int_{\mathcal{X}_a}\int_{\mathcal{X}_a}(\min(U,V)+1)dP_a dP_a)}.$$

The other parts of function `calc_Kv` *remain unchanged.*

(2) Calculate the Gram matrices K_v for all v:
First, for all $a = 1, ..., d$, the Gram matrices K_a associated with kernels k_{0a} are calculated using Equation "(2)" (see the Appendix (supplementary materials)), $(K_a)_{i,i'} = k_{0a}(X_{ai}, X_{ai'})$. Then, for all $v \in \mathcal{P}$, the Gram matrices K_v associated with kernel $k_v = \prod_{a\in v} k_{0a}$ are computed by $K_v = \bigodot_{a\in v} K_a$.

(3) Insure the positive definiteness of the matrices K_v:
The output of function `calc_Kv` is one of the input arguments of the functions associated with the RKHS group lasso and the RKHS ridge group sparse algorithms. Throughout these algorithms we need to calculate the inverse and the square root of the matrices K_v. In order to avoid the numerical problems and insure the invertibility of the matrices K_v, it is mandatory to have these matrices positive definite. One way to render the matrices K_v positive definite is to add a nugget effect to them. That is, to modify matrices K_v by adding a diagonal with a constant term, i.e. $K_v + \text{epsilon} \times I_n$. The value of epsilon is computed based on the data and through a part of the algorithm of the function `calc_kv`. Let us briefly explain this part of the algorithm:

For each group $v \in \mathcal{P}$, let $\lambda_{v,i}, i = 1, ..., n$ be the eigenvalues associated with the matrix K_v. Set $\lambda_{v,\max} = \max_i \lambda_{v,i}$ and $\lambda_{v,\min} = \min_i \lambda_{v,i}$. For some fixed value of tolerance `tol`, and for each matrix K_v, if "$\lambda_{v,\min} < \lambda_{v,\max} \times$ `tol`", then, the eigenvalues of K_v are replaced by $\lambda_{v,i}+$`epsilon`, with `epsilon` being equal to $\lambda_{v,\max} \times$ `tol`. The value of `tol` is set as $1e^{-8}$ by default, but one may consider a smaller or a greater value for it depending on the kernel chosen and the value of n.

The function `calc_Kv` is described in Section "Companion functions" (supplementary materials) and illustrated in Example 2.

Optimization algorithms

The RKHS meta-model is the solution of one of the optimization problems: the minimization of the RKHS group lasso criterion presented in Equation (14) (if $\gamma = 0$), or the minimization of the RKHS ridge group sparse criterion presented in Equation (13) (if $\gamma \neq 0$). In the following, the algorithms to solve these optimization problems are presented.

RKHS group lasso

A popular technique for doing group wise variable selection is group lasso. With this procedure, depending on the value of the tuning parameter μ, an entire group of predictors may drop out of the model. An efficient algorithm for solving group lasso problem is the classical block coordinate descent algorithm (Boyd et al., 2011; Bubeck, 2015). Following the idea of Fu (1998), Yuan and Lin (2006) implemented a block wise descent algorithm for the group lasso penalized least-squares under the condition that the model matrices in each group are orthonormal. A block coordinate (gradient) descent algorithm for solving the group lasso penalized logistic regression is then developed by Meier et al. (2008). This algorithm is implemented in the R package **grplasso** available from CRAN (Meier, 2020). Yang and Zou (2015) proposed a unified algorithm named group wise majorization descent for solving the general group lasso learning problems by assuming that the loss function satisfies a quadratic majorization condition. The implementation of their work is done in the **gglasso** R package available from CRAN (Yang et al., 2020).

In order to solve the RKHS group lasso optimization problem, we use the classical block coordinate descent algorithm. The minimization of criterion $C_g(f_0, \theta)$ (see Equation (14)) is done along each group v at a time. At each step of the algorithm, the criterion $C_g(f_0, \theta)$ is minimized as a function of the current block's parameters, while the parameters values for the other blocks are fixed to their current values. The procedure is repeated until convergence. This procedure leads to Algorithm 1 (see the Appendix (supplementary materials) for more details on this procedure). In the **RKHSMetaMod**

Algorithm 1 RKHS group lasso algorithm:

Set $\theta_0 = [0]_{|\mathcal{P}| \times n}$
repeat
  Calculate $f_0 = \text{argmin}_{f_0} C_g(f_0, \theta)$
  for $v \in \mathcal{P}$ **do**
    Calculate $R_v = Y - f_0 - \sum_{v \neq w} K_w \theta_w$
    if $2\|K_v^{1/2} R_v\| / \sqrt{n} \leq \mu_g$ **then**
      $\theta_v \leftarrow 0$
    else
      $\theta_v \leftarrow \text{argmin}_{\theta_v} C_g(f_0, \theta)$
    end if
  end for
until convergence

package, the Algorithm 1 is implemented as `RKHSgrplasso` function. This function is described in Section "Companion functions" (supplementary materials) and illustrated in Example 2.

RKHS ridge group sparse

In order to solve the RKHS ridge group sparse optimization problem, we propose an adapted block coordinate descent algorithm. This algorithm is provided in two steps:

Step 1 Initialize the input parameters by the solutions of the RKHS group lasso algorithm for each value of the tuning parameter μ, and implement the RKHS ridge group sparse algorithm through the

active support of the RKHS group lasso solutions until it achieves convergence. This step is provided in order to decrease the execution time. In fact, instead of implementing the RKHS ridge group sparse algorithm over the set of all groups $\mathcal{P}$, it is implemented only over the groups in the support of the solution of the RKHS group lasso algorithm, $\widehat{S}_{\widehat{f}_{\text{Group Lasso}}}$.

Step 2 Re-initialize the input parameters with the obtained solutions of Step 1 and implement the RKHS ridge group sparse algorithm through all groups in $\mathcal{P}$ until it achieves convergence. This second step makes it possible to verify that no group is missing in the output of Step 1.

This procedure leads to Algorithm 2 (see the Appendix (supplementary materials) for more details on this procedure). In the **RKHSMetaMod** package the Algorithm 2 is implemented as `pen_MetMod`

Algorithm 2 RKHS ridge group sparse algorithm:

Step 1:
Set $\theta_0 = \widehat{\theta}_{\text{Group Lasso}}$ and $\widehat{\mathcal{P}} = \widehat{S}_{\widehat{f}_{\text{Group Lasso}}}$
repeat
  Calculate $f_0 = \text{argmin}_{f_0} C(f_0, \theta)$
  for $v \in \widehat{\mathcal{P}}$ **do**
    Calculate $R_v = Y - f_0 - \sum_{v \neq w} K_w \theta_w$
    Solve $J^* = \text{argmin}_{\widehat{t}_v \in \mathbb{R}^n} \{J(\widehat{t}_v),$ such that $\|K_v^{-1/2} \widehat{t}_v\| \leq 1\}$
    if $J^* \leq \gamma$ **then**
      $\theta_v \leftarrow 0$
    else
      $\theta_v \leftarrow \text{argmin}_{\theta_v} C(f_0, \theta)$
    end if
  end for
until convergence
Step 2:
Implement the same procedure as **Step 1** with $\theta_0 = \widehat{\theta}_{\text{old}}$, $\widehat{\mathcal{P}} = \mathcal{P} \triangleright \widehat{\theta}_{\text{old}}$ is the estimation of θ in **Step 1**.

function. This function is described in Section "Companion functions" (supplementary materials) and illustrated in Example 2.

RKHS meta-model with at most $qmax$ groups in its support

By considering some prior information about the data, one may be interested in a RKHS meta-model $\widehat{f}$ with the number of groups in its support not greater than some "$qmax$". In order to obtain such an estimator, we provide the following procedure in the **RKHSMetaMod** package:

- First, the tuning parameter γ is set as zero and a value of μ for which the solution of the RKHS group lasso algorithm, Algorithm 1, contains exactly $qmax$ groups in its support is computed. This value is denoted by μ_{qmax}.
- Then, the RKHS ridge group sparse algorithm, Algorithm 2, is implemented by setting the tuning parameter μ equal to μ_{qmax} and a grid of values of the tuning parameter $\gamma > 0$.

This procedure leads to Algorithm 3. This algorithm is implemented in the **RKHSMetaMod** package, as function `RKHSMetMod_qmax` (see Section "Main RKHSMetaMod functions" (supplementary materials) for more details on this function).

Remark 4 *As both terms in the penalty function of criterion (13) enforce sparsity to the solution, the estimator obtained by solving the RKHS ridge group sparse associated with the pair of the tuning parameters $(\mu_{qmax}, \gamma > 0)$ may contain a smaller number of groups than the solution of the RKHS group lasso optimization problem (i.e. the RKHS ridge group sparse with $(\mu_{qmax}, \gamma = 0)$). And therefore, the estimated RKHS meta-model contains at most "qmax" groups in its support.*

4 RKHSMetaMod through examples

Simulation study

Let us consider the g-function of Sobol (Saltelli et al., 2009) in the Gaussian regression framework, i.e. $Y = m(X) + \varepsilon$. The error term ε is a centered Gaussian random variable with variance σ^2, and m is the

Algorithm 3 Algorithm to estimate RKHS meta-model with at most *qmax* groups in its support:

1: Calculate $\mu_{\max} = \max_v 2\|K_v^{1/2}(Y - \overline{Y})\|/\sqrt{n}$
2: Set $\mu_1 = \mu_{\max}$ and $\mu_2 = \mu_{\max}$/`rat` ▷ `rat` is setted by user.
3: **repeat**
4: Implement RKHS group lasso algorithm, Algorithm 1, with $\mu_i = (\mu_1 + \mu_2)/2$
5: Set $q = |\widehat{S}_{\widehat{f}_{\text{Group Lasso}}}|$
6: **if** $q > qmax$ **then**
7: Set $\mu_1 = \mu_1$ and $\mu_2 = \mu_i$
8: **else**
9: Set $\mu_1 = \mu_i$ and $\mu_2 = \mu_2$
10: **end if**
11: **until** $q = qmax$ or $i >$ `Num` ▷ `Num` is setted by user.
12: Implement RKHS ridge group sparse algorithm, Algorithm 2, with $(\mu = \mu_{q_{max}}, \gamma > 0)$

g-function of Sobol defined over $[0,1]^d$ by,

$$m(X) = \prod_{a=1}^{d} \frac{|4X_a - 2| + c_a}{1 + c_a},\ c_a > 0. \tag{17}$$

The Sobol indices of the g-function can be expressed analytically:

$$\forall v \in \mathcal{P},\ S_v = \frac{1}{D}\prod_{a\in v} D_a,\ D_a = \frac{1}{3(1+c_a)^2},\ D = \prod_{a=1}^{d}(D_a + 1) - 1.$$

Set $c_1 = 0.2$, $c_2 = 0.6$, $c_3 = 0.8$ and $(c_a)_{a>3} = 100$. With these values of coefficients c_a, the variables X_1, X_2 and X_3 explain 99.98% of the variance of function $m(X)$ (see Table 4).

In this Section, three examples are presented. In all examples, the value of Dmax is set as three. Example 1 illustrates the use of the `RKHSMetMod` function by considering three different kernels, `"matern"`, `"brownian"`, and `"gaussian"` (see Table 1), and three datasets of $n \in \{50, 100, 200\}$ observations and $d = 5$ input variables. The larger datasets with $n \in \{1000, 2000, 5000\}$ observations and $d = 10$ input variables are studied in Examples 2 and 3. In each example, two independent datasets are generated: (X, Y) to estimate the meta-models, and (XT, YT) to estimate the prediction errors. The design matrices X and XT are the Latin Hypercube Samples of the input variables that are generated using `maximinLHS` function of the package **lhs** available at CRAN (Carnell, 2021):

```
library(lhs); X <- maximinLHS(n, d); XT <- maximinLHS(n, d)
```

The response variables Y and YT are calculated as $Y = m(X) + \varepsilon$ and $YT = m(XT) + \varepsilon_T$, where ε, and ε_T are centered Gaussian random variables with $\sigma^2 = (0.2)^2$:

```
a <- c(0.2, 0.6, 0.8, 100, 100, 100, 100, 100, 100, 100)[1:d]
g=1; for (i in 1:d) g = g*(abs(4*X[,i]-2)+a[i])/(1+a[i])
sigma <- 0.2
epsilon <- rnorm(n, 0, sigma^2); Y <- g + epsilon
gT=1; for (i in 1:d) gT = gT*(abs(4*XT[,i]-2)+a[i])/(1+a[i])
epsilonT <- rnorm(n, 0, sigma^2); YT <- gT + epsilonT
```

Example 1 *RKHS meta-model estimation using* `RKHSMetMod` *function:*

In this example, three datasets of n points `maximinLHS` over $[0,1]^d$ with $n \in \{50, 100, 200\}$ and $d = 5$ are generated, and a grid of five values of tuning parameters μ and γ is considered as follows:

$$\mu_{(1:5)} = \mu_{max}/(\sqrt{n} \times 2^{(2:6)}), \quad \gamma_{(1:5)} = (0.2, 0.1, 0.01, 0.005, 0).$$

For each dataset, the experiment is repeated $N_r = 50$ times. At each repetition, the RKHS meta-models associated with the pair of tuning parameters (μ, γ) are estimated using the `RKHSMetMod` function:

```
Dmax <- 3; kernel <- "matern" # kernel <- "brownian" # kernel <- "gaussian"
gamma <- c(0.2, 0.1, 0.01, 0.005, 0); frc <- 1/(0.5^(2:6))
res <- RKHSMetMod(Y, X, kernel, Dmax, gamma, frc, FALSE)
```

These meta-models are evaluated using a testing dataset. The prediction errors are computed for them using the `PredErr` function. The RKHS meta-model with minimum prediction error is chosen to be the

best estimator for the model. Finally, the Sobol indices are computed for the *best* RKHS meta-model using the function `SI_emp`:

```
Err <- PredErr(X, XT, YT, mu, gamma, res, kernel, Dmax)
SI <- SI_emp(res, Err)
```

The vector `mu` is the values of the tuning parameter μ that are calculated throughout the function `RKHSMetMod`. It could be recovered from the output of the `RKHSMetMod` function as follows:

```
mu <- vector()
l <- length(gamma); for(i in 1:length(frc)){mu[i] <- res[[(i-1)*l+1]]$mu}
```

The performance of this method for estimating a meta-model is evaluated by considering a third dataset $(m(X_i^{third}), X_i^{third})$, $i = 1, ..., N$, with $N = 1000$. The global prediction error is calculated as follows:

Let $\widehat{f}_r(.)$ be the *best* RKHS meta-model obtained in the repetition r, $r = 1, ..., N_r$, then

$$GPE = \frac{1}{N_r}\sum_{r=1}^{N_r}\frac{1}{N}\sum_{i=1}^{N}(\widehat{f}_r(X_i^{third}) - m(X_i^{third}))^2.$$

The values of GPE obtained for different kernels and values of n are given in Table 2. As expected

n	50	100	200
GPE_m	0.13	0.07	0.03
GPE_b	0.14	0.10	0.05
GPE_g	0.15	0.10	0.07

Table 2: Example 1: The columns of the table correspond to the different datasets with $n \in \{50, 100, 200\}$ and $d = 5$. Each line of the table, from up to down, gives the value of GPE obtained for each dataset associated with the "`matern`", "`brownian`" and "`gaussian`" kernels, respectively.

the value of GPE decreases as n increases. The lowest values of GPE are obtained when using the "`matern`" kernel.

In order to sum up the behaviour of the procedure for estimating the Sobol indices, we consider the mean square error (MSE) criterion obtained by $\sum_v(\sum_{r=1}^{N_r}(\widehat{S}_{v,r} - S_v)^2/N_r)$, where for each group v, S_v denotes the true values of the Sobol indices, and $\widehat{S}_{v,r}$ is the empirical Sobol indices of the *best* RKHS meta-model in repetition r. The obtained values of MSE for different kernels and values of n are given in Table 3. As expected, the values of MSE are smaller for larger values of n. The smallest values are

n	50	100	200
MSE_m	75.12	46.72	28.22
MSE_b	110.71	84.99	41.06
MSE_g	78.22	94.67	67.02

Table 3: Example 1: The columns of the table correspond to the different datasets with $n \in \{50, 100, 200\}$ and $d = 5$. Each line of the table, from up to down, gives the value of MSE obtained for each dataset associated with the "`matern`", "`brownian`" and "`gaussian`" kernels, respectively.

obtained when using the "`matern`" kernel.

The means of the empirical Sobol indices of the *best* RKHS meta-models through all repetitions for $n = 200$ and "`matern`" kernel are displayed in Table 4. It appears that the estimated Sobol indices

v	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$	sum
S_v	43.30	24.30	19.20	5.63	4.45	2.50	0.57	99.98
$\widehat{S}_{v,.}$	46.10	26.33	20.62	2.99	2.22	1.13	0.0	99.39

Table 4: Example 1: The first line of the table gives the true values of the Sobol indices $\times 100$. The second line gives the mean of the estimated empirical Sobol indices $(\widehat{S}_{v,.} = \sum_{r=1}^{N_r}\widehat{S}_{v,r}/N_r) \times 100$ greater than 10^{-2} calculated over fifty simulations for $n = 200$ and "`matern`" kernel. The sum of the Sobol indices is displayed in the last column.

are close to the true ones, nevertheless, they are overestimated for the main effects, i.e. groups $v \in \{\{1\}, \{2\}, \{3\}\}$, and underestimated for the interactions of order two and three, i.e. groups $v \in \{\{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$. Note that the strategy of choosing the tuning parameters is based

on the minimization of the prediction error of the estimated meta-model, which may not minimize the error of estimating the Sobol indices.

Taking into account the results obtained for this Example 1, the calculations in the rest of the examples is done using only the "matern" kernel.

Example 2 *A time-efficient strategy to obtain the "optimal" tuning parameters when dealing with large datasets:*

A dataset of n points maximinLHS over $[0,1]^d$ with $n = 1000$ and $d = 10$ is generated. First, we use functions calc_Kv and mu_max to compute the eigenvalues and eigenvectors of the positive definite matrices K_v, and the value of μ_{max}, respectively:

```
kernel <- "matern"; Dmax <- 3
Kv <- calc_Kv(X, kernel, Dmax, TRUE, TRUE)
mumax <- mu_max(Y, Kv$kv)
```

Then, we consider the two following steps:

1. Set $\gamma = 0$ and, $\mu_{(1:9)} = \mu_{max}/(\sqrt{n} \times 2^{(2:10)})$. Calculate the RKHS meta-models associated with the values of $\mu_g = \mu \times \sqrt{n}$ by using the function RKHSgrplasso. Gather the obtained RKHS meta-models in a list, res_g (while this job could be done using the function RKHSMetMod by setting $\gamma = 0$, in this example, we use the function RKHSgrplasso in order to avoid the re-calculation of K_v's at the next step). Thereafter, for each estimator in res_g, the prediction error is calculated by considering a new dataset and using the function PredErr. The value of μ with the smallest error of prediction in this step is denoted by μ_i. Let us implement this step:
   For a grid of values of μ_g, a sequence of the RKHS meta-models are calculated and gathered in the res_g list:

   ```
   mu_g <- c(mumax*0.5^(2:10))
   res_g <- list(); resg <- list()
   for(i in 1:length(mu_g)){
     resg[[i]] <- RKHSgrplasso(Y, Kv, mu_g[i], 1000, FALSE)
     res_g[[i]] <- list("mu_g"=mu_g, "gamma"=0, "MetaModel"=resg[[i]])
   }
   ```

   Output res_g contains nine RKHS meta-models and they are evaluated using a testing dataset:

   ```
   gamma <- c(0); Err_g <- PredErr(X, XT, YT, mu_g, gamma, res_g, kernel, Dmax)
   ```

   The prediction errors of the RKHS meta-models obtained in this step are displayed in Table 5. It appears that the minimum prediction error corresponds to the solution of the RKHS group lasso algorithm with $\mu_g = 0.041$, so $\mu_i = 0.041/\sqrt{n}$.

μ_g	1.304	0.652	0.326	0.163	0.081	0.041	0.020	0.010	0.005
$\gamma = 0$	0.197	0.156	0.145	0.097	0.063	0.055	0.056	0.063	0.073

Table 5: Example 2: Prediction errors associated with the RKHS meta-models computed in step 1.

2. Choose a smaller grid of values of μ, $(\mu_{(i-1)}, \mu_i, \mu_{(i+1)})$, and set a grid of values of $\gamma > 0$. Calculate the RKHS meta-models associated with each pair of the tuning parameters (μ, γ) by the function pen_MetMod. Calculate the prediction errors for the new sequence of the RKHS meta-models using the function PredErr. Compute the empirical Sobol indices for the *best* estimator. Let us go back to the implementation of the example and apply this step 2:
   The grid of the values of μ in this step is $(0.081, 0.041, 0.020)/\sqrt{n}$. The RKHS meta-models associated with this grid of the values of μ are gathered in a new list resgnew. We set $\gamma_{(1:4)} = (0.2, 0.1, 0.01, 0.005)$, and we calculate the RKHS meta-models for this new grid of the values of (μ, γ) using pen_MetMod function:

   ```
   gamma <- c(0.2, 0.1, 0.01, 0.005); mu <- c(mu_g[5], mu_g[6], mu_g[7])/sqrt(n)
   resgnew <- list()
   resgnew[[1]] <- resg[[5]]; resgnew[[2]] <- resg[[6]]; resgnew[[3]] <- resg[[7]]
   res <- pen_MetMod(Y, Kv, gamma, mu, resgnew, 0, 0)
   ```

   The output res is a list of twelve RKHS meta-models. These meta-models are evaluated using a new dataset, and their prediction errors are displayed in Table 6. The minimum prediction error is associated with the pair $(0.020/\sqrt{n}, 0.01)$, and the *best* RKHS meta-model is then $\widehat{f}_{(0.020/\sqrt{n}, 0.01)}$.

μ	$0.081/\sqrt{n}$	$0.041/\sqrt{n}$	$0.020/\sqrt{n}$
$\gamma = 0.2$	0.153	0.131	0.119
$\gamma = 0.1$	0.098	0.079	0.072
$\gamma = 0.01$	0.065	0.054	0.053
$\gamma = 0.005$	0.064	0.054	0.054

Table 6: Example 2: Obtained prediction errors in step 2.

The performance of this procedure for estimating the Sobol indices is evaluated using the relative error (RE) defined as follows:
For each v, let S_v be the true value of the Sobol indices displayed in Table 4 and $\widehat{S}_v$ be the estimated empirical Sobol indices. Then

$$RE = \sum_v |\widehat{S}_v - S_v| / S_v. \tag{18}$$

In Table 7 the estimated empirical Sobol indices, their sum, and the value of RE are displayed.

v	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$	sum	RE
$\widehat{S}_v$	42.91	25.50	20.81	4.40	3.84	2.13	0.00	99.60	1.64

Table 7: Example 2: The estimated empirical Sobol indices $\times 100$ greater than 10^{-2}. The last two columns show $\sum_v \widehat{S}_v$ and RE, respectively.

The obtained RE for each group v is smaller than 1.64%, therefore, the estimated Sobol indices in this example are very close to the true values of the Sobol indices displayed in the first row of Table 4.

Example 3 *Dealing with larger datasets:*

Two datasets of n points `maximinLHS` over $[0,1]^d$ with $n \in \{2000, 5000\}$ and $d = 10$ are generated. In order to obtain one RKHS meta-model associated with one pair of the tuning parameters (μ, γ), the number of coefficients to be estimated is equal to $n \times$vMax$= n \times 175$. Table 8 gives the execution time for different functions used throughout the Examples 1-3. In all examples we used a cluster of computers with: 2 Intel Xeon E5-2690 processors (2.90GHz) and 96Gb Ram (6x16Gb of memory 1600MHz). As we can see, the execution time increases fast as n increases. In Figure 1 the plot

(n,d)	`calc_Kv`	`mu_max`	`RKHSgrplasso`	`pen_MetMod`	$\lvert S_{\widehat{f}}\rvert$	sum
(100,5)	0.09s	0.01s	1s	2s	18	$\sim$ 3s
			2s	3s	19	$\sim$ 5s
(500,10)	33s	9s	247s	333s	39	$\sim$ 10min
			599s	816s	64	$\sim$ 24min
(1000,10)	197s	53s	959s	1336s	24	$\sim$ 42min
			2757s	4345s	69	$\sim$ 2h
(2000,10)	1498s	420s	3984s	4664s	12	$\sim$ 2h:56min
			12951s	22385s	30	$\sim$ 10h:20min
(5000,10)	34282s	6684s	38957s	49987s	11	$\sim$ 36h:05min
			99221s	111376s	15	$\sim$ 69h:52min

Table 8: Example 3: The kernel used is "`matern`". The execution time for the functions `RKHSgrplasso` and `pen_MetMod` is displayed in each row for two pairs of values of the tuning parameters $(\mu_1 = \mu_{max}/(\sqrt{n} \times 2^7), \gamma = 0.01)$ on up, and $(\mu_2 = \mu_{max}/(\sqrt{n} \times 2^8), \gamma = 0.01)$ on below. In the column $|S_{\widehat{f}}|$, the number of groups in the support of each estimated RKHS meta-model is displayed.

of the logarithm of the time (in seconds) versus the logarithm of n is displayed for the functions `calc_Kv`, `mu_max`, `RKHSgrplasso` and `pen_MetMod`. It appears that, the algorithms of these functions are of polynomial time $O(n^\alpha)$ with $\alpha \backsimeq 3$ for the functions `calc_Kv` and `mu_max`, and $\alpha \backsimeq 2$ for the functions `RKHSgrplasso` and `pen_MetMod`.

Taking into account the results obtained for the prediction error and the values of $(\widehat{\mu}, \widehat{\gamma})$ in Example 2, in this example, only two values of the tuning parameter $\mu_{(1:12)} = \mu_{max}/(\sqrt{n} \times 2^{(7:8)})$, and one value of the tuning parameter $\gamma = 0.01$ are considered. The RKHS meta-models associated with the pair of values (μ_i, γ), $i = 1, 2$ are estimated using the `RKHSMetMod` function:

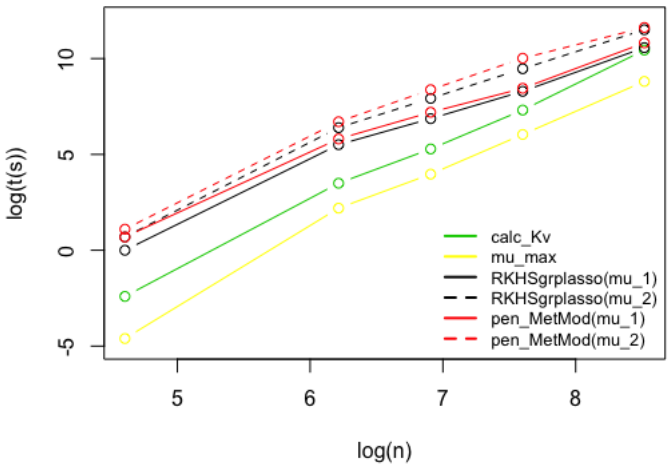


Figure 1: Example 3: Timing plot for $d = 10$, $n \in \{100, 300, 500, 1000, 2000, 5000\}$, and different functions of the **RKHSMetaMod** package. The logarithm of the execution time (in seconds) for the functions `RKHSgrplasso` and `pen_MetMod` is displayed for two pairs of values of the tuning parameters $(\mu_1 = \mu_{max}/(\sqrt{n} \times 2^7), \gamma = 0.01)$ in solid lines, and $(\mu_2 = \mu_{max}/(\sqrt{n} \times 2^8), \gamma = 0.01)$ in dashed lines.

```
kernel <- "matern"; Dmax <- 3
gamma <- c(0.01); frc <- 1/(0.5^(7:8))
res <- RKHSMetMod(Y, X, kernel, Dmax, gamma, frc, FALSE)
```

The prediction error and the empirical Sobol indices are then calculated for the obtained meta-models using the functions `PredErr` and `SI_emp`:

```
mu <- vector(); mu[1] <- res[[1]]$mu; mu[2] <- res[[2]]$mu
Err <- PredErr(X, XT, YT, mu, gamma, res, kernel, Dmax)
SI <- SI_emp(res, NULL)
```

Table 9 gives the estimated empirical Sobol indices as well as their sum, the values of RE (see Equation (18)), and the prediction errors associated with the obtained estimators. For $n = 5000$ we obtained the

n	v	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$	sum	RE	Err
2000	$\widehat{S}_{v;(\mu_1,\gamma)}$	45.54	24.78	21.01	3.96	3.03	1.65	0.00	99.97	2.12	0.052
	$\widehat{S}_{v;(\mu_2,\gamma)}$	45.38	25.07	19.69	4.36	3.66	1.79	0.00	99.95	1.79	0.049
5000	$\widehat{S}_{v;(\mu_1,\gamma)}$	44.77	25.39	20.05	4.49	3.38	1.90	0.00	99.98	1.81	0.049
	$\widehat{S}_{v;(\mu_2,\gamma)}$	43.78	24.99	19.56	5.43	3.90	2.32	0.00	99.98	1.29	0.047

Table 9: Example 3: The estimated empirical Sobol indices $\times 100$ greater than 10^{-2} associated with each estimated RKHS meta-model is printed. The last three columns show $\sum_v \widehat{S}_v$, RE, and the prediction error (Err), respectively. We have $\mu_1 = \mu_{max}/(\sqrt{n} \times 2^7)$, $\mu_2 = \mu_{max}/(\sqrt{n} \times 2^8)$ and $\gamma = 0.01$.

smaller values of RE and prediction error (Err). So as expected, the estimation of the Sobol indices as well as the prediction quality are better for larger values of n.

In Figure 2 the result of the estimation quality and the Sobol indices for the dataset with n equal to 5000, d equal to 10, and (μ_2, γ) are displayed. The line $y = x$ in red crosses the cloud of points as long as the values of the g-function are smaller than three. When the values of the g-function are greater than three, the estimator $\widehat{f}$ tends to underestimate the g-function. Concerning the Sobol indices obtained by the estimator $\widehat{f}$, as illustrated in the right-hand plot, with the exception of groups $\{1\}$, $\{2\}$, $\{3\}$, $\{1,2\}$, $\{1,3\}$, and $\{2,3\}$ for which we obtained significant values of the sobol indices, for all other groups the estimated sobol indices are very small and almost zero.

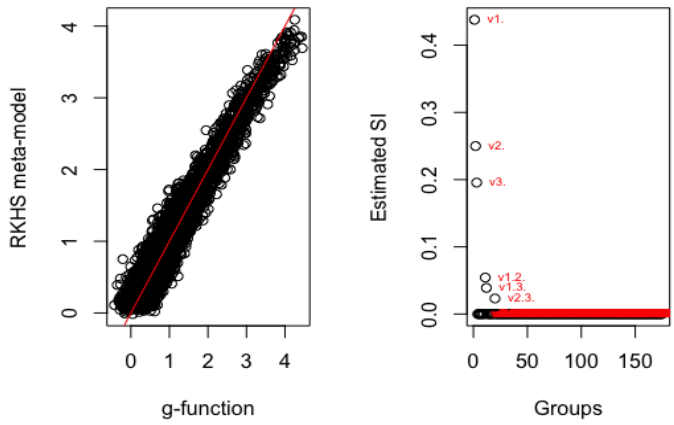


Figure 2: Example 3: On the left, the RKHS meta-model versus the g-function is plotted. On the right, the empirical Sobol indices in the y-axis and vMax= 175 groups in the x-axis are displayed.

Comparison examples

This section includes two examples. In the first example we reproduce an example from paper Gu et al. (2019) and compare the prediction quality of the RKHS meta-model with the GP (Kriging) based meta-models from the **RobusGaSP** (Gu et al., 2019) and **DiceKriging** (Roustant et al., 2012) packages. The objective is to evaluate the quality of the RKHS meta-model and to compare it with methods recently proposed for approximating complex models. In the first example we consider one-dimensional model and focus on the comparison between the true model and the estimated meta-model. In the second example we reproduce an example from paper Roustant et al. (2012) which allows us to compare the prediction quality of the RKHS meta-model with the Kriging based meta-model from **DiceKriging** package, as well as the estimation quality of the Sobol indices in our package with the well-known package **sensitivity**. For the sake of comparison between the three methods, the meta-models are calculated using the same experimental design and outputs, and the same kernel function available in three packages is used. However, in packages **RobustGaSP** and **DiceKriging** the range parameter r (see Table 1) in the kernel function is estimated by marginal posterior modes with the robust parameterization and by MLE with upper and lower bounds, respectively, while it is assumed to be fixed and set as $\sqrt{3}/2$ in the **RKHSMetaMod** package.

Example 4 *"The modified sine wave function":*

We consider the 1-dimensional modified sine wave function defined by $m(X) = 3\sin(5\pi X) + \cos(7\pi X)$ over $[0,1]$. The same experimental design as described in Gu et al. (2019) is considered: the design matrix X is a sequence of 12 equally spaced points on $[0,1]$, and the response variable Y is calculated as $Y = m(X)$:

```
X <- as.matrix(seq(0,1,1/11)); Y <- sinewave(X)
```

where `sinewave` function is defined in Gu et al. (2019). We build the GP based meta-models by the **RobustGaSP** and the **DiceKriging** packages using the constant mean function and kernel Matérn $3/2$:

```
library(RobustGaSP)
res.rgasp <- rgasp(design=X, response=Y, kernel_type="matern_3_2")
library(DiceKriging)
res.km <- km(design=X, response=Y, covtype="matern3_2")
```

As $d = 1$, we have $Dmax = 1$. We consider the grid of values of $\mu_{(1:9)} = \mu_{max}/(\sqrt{n} \times 2^{(2:10)})$ and $\gamma_{(1:5)} = (0.2, 0.1, 0.01, 0.005, 0)$. The RKHS meta-models associated with the pair of values (μ_i, γ_j), $i = 1, \cdots, 9$, $j = 1, \cdots, 5$ are estimated using the `RKHSMetMod` function:

```
kernel <- "matern"; Dmax <- 1
gamma <- c(0.2, 0.1, 0.01, 0.005,0); frc <- 1/(0.5^(2:10))
res <- RKHSMetMod(Y, X, kernel, Dmax, gamma, frc, FALSE)
```

Given a testing dataset (XT, YT), the prediction errors associated with the obtained RKHS meta-models are calculated using the `PredErr` function, and the *best* RKHS meta-model is chosen to be the estimator of the model $m(X)$:

```
XT <- as.matrix(seq(0,1,1/11)); YT <- sinewave(XT)
Err <- PredErr(X, XT, YT, mu, gamma, res, kernel, Dmax)
```

To compare these three estimators in terms of the prediction quality, we perform prediction on 100 test points, equally spaced in $[0, 1]$:

```
predict_X <- as.matrix(seq(0,1,1/99))
#prediction with the GP based meta-models:
rgasp.predict <- predict(res.rgasp, predict_X)
km.predict <- predict(res.km, predict_X, type='UK')
#prediction with the best RKHS meta-model:
res.predict <- prediction(X, predict_X, kernel, Dmax, res, Err)
```

The prediction results are plotted in Figure 3. The black circles that correspond to the prediction from the **RKHSMetMod** package are closer to the real output than the green and the blue circles corresponding to the predictive means from the **RobustGaSP** and **DiceKriging** packages. The meta-

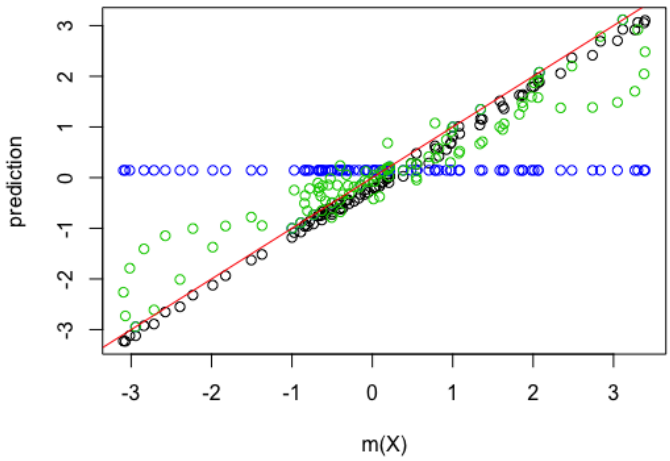


Figure 3: Example 4: Prediction of the modified sine wave function with 100 test points, equally spaced in $[0, 1]$. The x-axis is the real output and the y-axis is the prediction. The black circles are the prediction from **RKHSMetMod**, the green circles are the predictive mean from **RobustGaSP**, and the blue circles are the predictive mean from **DiceKriging**.

model results are plotted in Figure 4. The prediction from the **RKHSMetaMod** package plotted as the black curve is much more accurate as an estimate of the true function (plotted in red) than the predictive mean from the **RobustGaSP** and **DiceKriging** packages plotted as the blue and green curves, respectively. As already noted by Gu et al. (2019), for that sine wave example, the meta-model from the DiceKriging package "degenerates to the fitted mean with spikes at the design points".

Example 5 *"A standard SA 8-dimensional example":*

We consider the 8-dimensional g-function of Sobol implemented in the package **sensitivity**: the function $m(X)$ as defined in Equation (17) with coefficients $c_1 = 0$, $c_2 = 1$, $c_3 = 4.5$, $c_4 = 9$, $(c_a)_{a=5,6,7,8} = 99$. With these values of coefficients c_a, the variables X_1, X_2, X_3 and X_4 explain 99.96% of the variance of function $m(X)$ (see Table 10).

We consider the same experimental design as described in Roustant et al. (2012): the design matrices X and XT are 80-point optimal Latin Hypercube Samples of the input variables generated by the `optimumLHS` function of package **lhs**, and the response variables Y and YT are calculated as $Y = m(X)$, and $YT = m(XT)$ using `sobol.fun` function of the package **sensitivity**:

```
n <- 80; d <- 8
library(lhs); X <- optimumLHS(n, d); XT <- optimumLHS(n, d)
library(sensitivity); Y <- sobol.fun(X); YT <- sobol.fun(XT)
```

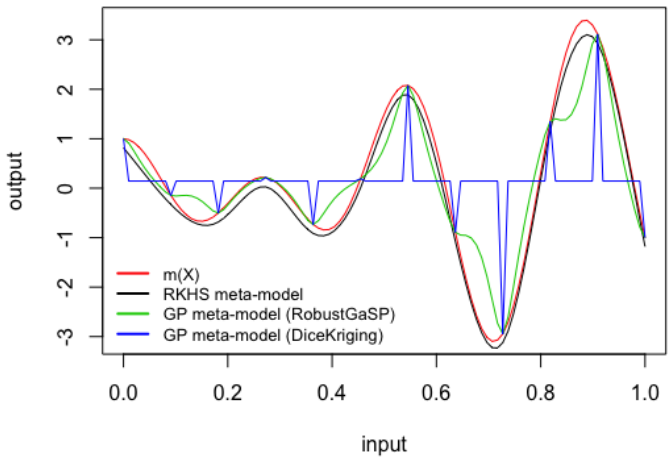


Figure 4: Example 4: The red curve is the graph of the modified sine wave function with 100 test points, equally spaced in $[0,1]$. The black curve is the prediction produced by the **RKHSMetaMod** package. The blue curve is the predictive mean by the **DiceKriging** package, and the green curve is the predictive mean produced by the **RobustGaSP** package.

Let us first consider the RKHS meta-model method. We set Dmax= 3, and we consider the grid of values of $\mu_{(1:9)} = \mu_{max}/(\sqrt{n} \times 2^{(2:10)})$, and $\gamma_{(1:5)} = (0.2, 0.1, 0.01, 0.005, 0)$. The RKHS meta-models associated with the pair of values $(\mu_i, \gamma_j), i = 1, \cdots, 9, j = 1, \cdots, 5$ are estimated using the `RKHSMetMod` function:

```
kernel <- "matern"; Dmax <- 3
gamma <- c(0.2, 0.1, 0.01, 0.005,0); frc <- 1/(0.5^(2:10))
res <- RKHSMetMod(Y, X, kernel, Dmax, gamma, frc, FALSE)
```

Given the testing dataset (XT, YT), the prediction errors associated with the obtained RKHS meta-models are calculated using `PredErr` function, and the *best* RKHS meta-model is chosen to be the estimator of the model $m(X)$. Finally, the Sobol indices are computed for the *best* RKHS meta-model using the function `SI_emp`:

```
Err <- PredErr(X, XT, YT, mu, gamma, res, kernel, Dmax)
SI <- SI_emp(res, Err)
```

Secondly, let us build the GP based meta-model. We use the `km` function of the package **DiceKriging** with the constant mean function and kernel Matérn 3/2:

```
library(DiceKriging)
res.km <- km(design = X, response = Y, covtype = "matern3_2")
```

The Sobol indices associated with the estimated GP based meta-model are calculated using `fast99` function of the package **sensitivity**:

```
SI.km <- fast99(model = kriging.mean, factors = d, n = 1000,
                      q = "qunif", q.arg = list(min = 0, max = 1), m = res.km)
```

where `kriging.mean` function is defined in Roustant et al. (2012).

The result of the estimation with the *best* RKHS meta-model and the Kriging based meta-model is drawn in Figure 5. The black circles that correspond to the *best* RKHS meta-model are closer to the real output than the blue circles corresponding to the GP based meta-model from the **DiceKriging** package. Another way to evaluate the prediction quality of the estimated meta-models is to consider the mean square error of the fitted meta-model computed by $\sum_{i=1}^{80}(m(X_i) - \widehat{f}(X_i))^2/80$. We obtained 3.96% and 0.07% for the Kriging based meta-model and the RKHS meta-model, respectively, which confirms the good behavior of the RKHS meta-model.

The estimated Sobol indices associated with the RKHS meta-model and the Kriging based meta-model are given in Table 10. As shown, with RKHS meta-model, we obtained non-zero values for the interactions of order two. Concerning the main effects, excepting the first one, the estimated Sobol

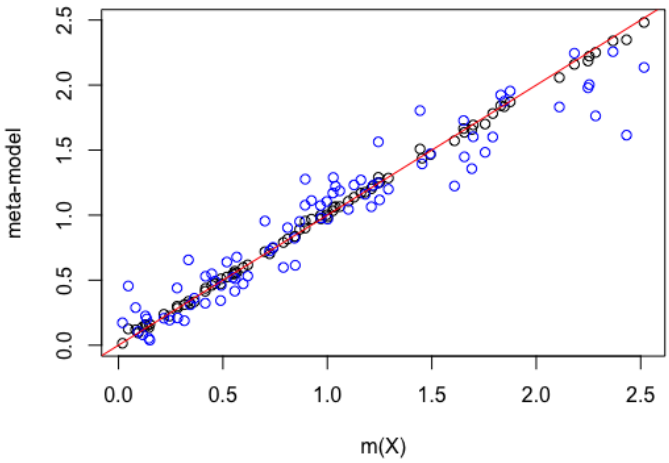


Figure 5: Example 5: The x-axis is the real output and the y-axis is the fitted meta-model. The black circles are the meta-model from **RKHSMetMod** and the blue circles are the meta-model from **DiceKriging**.

v	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{1,2\}$	$\{1,3\}$	$\{1,4\}$	$\{2,3\}$	$\{2,4\}$	$\{1,2,3\}$	$\{1,2,4\}$	sum
S_v	71.62	17.90	2.37	0.72	5.97	0.79	0.24	0.20	0.06	0.07	0.02	99.96
$\widehat{S}_v$	75.78	17.42	1.71	0.47	4.00	0.05	0.07	0.28	0.09	0.00	0.00	99.87
$\widehat{S}_{km_v}$	71.18	15.16	1.42	0.44	0.00	0.00	0.00	0.00	0.00	0.00	0.00	88.20

Table 10: Example 5: The true values of the Sobol indices $\times 100$ greater than 10^{-2} are given in the first raw. The estimated Sobol indices associated with the RKHS meta-model ($\widehat{S}_v$) and the Kriging based meta-model ($\widehat{S}_{km_v}$) are given in second and third rows, respectively.

indices with the RKHS meta-model are closer to the true ones. However, the interactions of order three are ignored by both meta-models. For a general comparison of the estimation quality of the Sobol indices, one may consider the criterion RE defined in Equation (18), which is equal to 7.95 for the Kriging based meta-model, and 5.59 for the RKHS meta-model. Comparing the values of RE, we can point out that the Sobol indices are better estimated with the RKHS meta-model in that model.

5 Summary and discussion

In this paper, we proposed an R package, called **RKHSMetaMod**, that estimates a meta-model of a complex model m. This meta-model belongs to a reproducing kernel Hilbert space constructed as a direct sum of Hilbert spaces (Durrande et al., 2013). The estimation of the meta-model is carried out via a penalized least-squares minimization allowing both to select and estimate the terms in the Hoeffding decomposition, and therefore, to select the Sobol indices that are non-zero and estimate them (Huet and Taupin, 2017). This procedure makes it possible to estimate the Sobol indices of high order, a point known to be difficult in practice. Using the convex optimization tools, **RKHSMetaMod** package implements two optimization algorithms: the minimization of the RKHS ridge group sparse criterion (13) and the RKHS group lasso criterion (14). Both of these algorithms rely on the Gram matrices K_v, $v \in \mathcal{P}$ and their positive definiteness. Currently, the package considers only uniformly distributed input variables. If one is interested by another distribution of the input variables, it suffices to modify the calculation of the kernels k_{0a}, $a = 1, ..., d$ in the function `calc_Kv` of this package (see Remark 3). The available kernels in the **RKHSMetaMod** package are: Gaussian kernel (with the fixed range parameter $r = 1/2$), Matérn kernel (with the fixed range parameter $r = \sqrt{3}/2$), Brownian kernel, quadratic kernel and linear kernel (see Table 1). With regard to the problem being under study, one may consider other kernels or kernels with different values of the range parameter r and add them easily to the list of the kernels in the `calc_Kv` function. For the large values of n and d the calculation and storage of eigenvalues and eigenvectors of all the Gram matrices K_v, $v \in \mathcal{P}$ require a lot of time and a very large amount of memory. In order to optimize the execution time and also

the storage memory, except for a function that is written in R, all of the functions of **RKHSMetaMod** package are written using the efficient C++ libraries through **RcppEigen** and **RcppGSL** packages. These functions are then interfaced with the R environment in order to contribute a user friendly package.

Bibliography

G. Blatman and B. Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of computational Physics*, 230:2345–2367, 2011. [p102]

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011. ISSN 1935-8237. doi: 10.1561/2200000016. URL https://doi.org/10.1561/2200000016. [p109]

S. Bubeck. Convex Optimization: Algorithms and Complexity. *Found. Trends Mach. Learn.*, 8(3-4): 231–357, Nov. 2015. ISSN 1935-8237. doi: 10.1561/2200000050. URL https://doi.org/10.1561/2200000050. [p109]

R. Carnell. *lhs: Latin Hypercube Samples*, 2021. URL https://CRAN.R-project.org/package=lhs. R package version 1.1.3. [p111]

G. M. Dancik and K. S. Dorman. mlegp: statistical analysis for computer models of biological systems using R. *Bioinformatics*, 24(17):1966, 2008. [p104]

N. Durrande, D. Ginsbourger, and O. Roustant. Additive Covariance kernels for high-dimensional Gaussian Process modeling. *Annales de la faculté des sciences de Toulouse Mathématiques*, 21(3):481–499, 4 2012. URL http://eudml.org/doc/251000. [p102]

N. Durrande, D. Ginsbourger, O. Roustant, and L. Carraro. ANOVA kernels and RKHS of zero mean functions for model-based sensitivity analysis. *Journal of Multivariate Analysis*, 115:57 – 67, 2013. ISSN 0047-259X. doi: https://doi.org/10.1016/j.jmva.2012.08.016. URL http://www.sciencedirect.com/science/article/pii/S0047259X1200214X. [p101, 103, 108, 119]

W. J. Fu. Penalized Regressions: The Bridge versus the Lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998. ISSN 10618600. URL http://www.jstor.org/stable/1390712. [p109]

M. Gu. Jointly Robust Prior for Gaussian Stochastic Process in Emulation, Calibration and Variable Selection. *Bayesian Analysis*, 14(3):857 – 885, 2019. doi: 10.1214/18-BA1133. URL https://doi.org/10.1214/18-BA1133. [p104]

M. Gu, X. Wang, and J. O. Berger. Robust Gaussian stochastic process emulation. *The Annals of Statistics*, 46(6A):3038 – 3066, 2018. doi: 10.1214/17-AOS1648. URL https://doi.org/10.1214/17-AOS1648. [p104]

M. Gu, J. Palomo, and O. Berger, James. RobustGaSP: Robust Gaussian Stochastic Process Emulation in R. *The R Journal*, 11(1):112, 2019. ISSN 2073-4859. doi: 10.32614/rj-2019-011. URL http://dx.doi.org/10.32614/rj-2019-011. [p104, 105, 116, 117]

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015. ISBN 1498712169, 9781498712163. [p103]

T. Homma and A. Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering & System Safety*, 52(1):1 – 17, 1996. ISSN 0951-8320. doi: https://doi.org/10.1016/0951-8320(96)00002-6. URL http://www.sciencedirect.com/science/article/pii/0951832096000026. [p102]

S. Huet and M.-L. Taupin. Metamodel construction for sensitivity analysis. *ESAIM: Procs*, 60:27–69, 2017. doi: 10.1051/proc/201760027. URL https://doi.org/10.1051/proc/201760027. [p101, 103, 119]

H. Kamari. *RKHSMetaMod: Ridge Group Sparse Optimization Problem for Estimation of a Meta Model Based on Reproducing Kernel Hilbert Spaces*, 2019. URL https://CRAN.R-project.org/package=RKHSMetaMod. R package version 1.1. [p105]

M. C. Kennedy and A. O'Hagan. Bayesian Calibration of Computer Models. *Journal of the Royal Statistical Society, Series B, Methodological*, 63:425–464, 2000. [p104]

G. S. Kimeldorf and G. Wahba. A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines. *Ann. Math. Statist.*, 41(2):495–502, 04 1970. doi: 10.1214/aoms/1177697089. URL http://dx.doi.org/10.1214/aoms/1177697089. [p105]

J. P. C. Kleijnen. *Design and Analysis of Simulation Experiments*. Springer Publishing Company, Incorporated, 1st edition, 2007. ISBN 0387718125, 9780387718125. [p102]

J. P. C. Kleijnen. Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, 192(3):707 – 716, 2009. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2007.10.013. URL http://www.sciencedirect.com/science/article/pii/S0377221707010090. [p102]

L. Le Gratiet, C. Cannamela, and B. Iooss. A Bayesian Approach for Global Sensitivity Analysis of (Multifidelity) Computer Codes. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):336–363, 2014. doi: 10.1137/130926869. URL https://doi.org/10.1137/130926869. [p102]

L. Le Gratiet, S. Marelli, and B. Sudret. *Metamodel-Based Sensitivity Analysis: Polynomial Chaos Expansions and Gaussian Processes*, pages 1289–1325. Springer International Publishing, Cham, 2017. ISBN 978-3-319-12385-1. doi: 10.1007/978-3-319-12385-1_38. URL https://doi.org/10.1007/978-3-319-12385-1_38. [p103]

Y. Lin and H. H. Zhang. Component selection and smoothing in multivariate nonparametric regression. *Ann. Statist.*, 34(5):2272–2297, 10 2006. doi: 10.1214/009053606000000722. URL https://doi.org/10.1214/009053606000000722. [p103]

H. Liu, L. Wasserman, and J. D. Lafferty. Nonparametric regression and classification with joint sparsity constraints. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 969–976. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3616-nonparametric-regression-and-classification-with-joint-sparsity-constraints.pdf. [p103]

A. Marrel, B. Iooss, B. Laurent, and O. Roustant. Calculations of Sobol indices for the Gaussian process metamodel. *Reliability Engineering & System Safety*, 94(3):742 – 751, 2009. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2008.07.008. URL http://www.sciencedirect.com/science/article/pii/S0951832008001981. [p102]

W. Mebane and J. Sekhon. Genetic Optimization Using Derivatives: The rgenoud Package for R. *Journal of Statistical Software, Articles*, 42(11):1–26, 2011. ISSN 1548-7660. doi: 10.18637/jss.v042.i11. URL https://www.jstatsoft.org/v042/i11. [p104]

L. Meier. *grplasso: Fitting User-Specified Models with Group Lasso Penalty*, 2020. URL https://CRAN.R-project.org/package=grplasso. R package version 0.4-7. [p109]

L. Meier, S. van de Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society. Series B*, 70(1):53–71, 2008. doi: 10.1111/j.1467-9868.2007.00627.x. [p109]

L. Meier, S. van de Geer, and P. Bühlmann. High-dimensional additive modeling. *Ann. Statist.*, 37(6B):3779–3821, 12 2009. doi: 10.1214/09-AOS692. URL https://doi.org/10.1214/09-AOS692. [p103]

J. E. Oakley and A. O'Hagan. Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(3):751–769, 2004. doi: 10.1111/j.1467-9868.2004.05304.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2004.05304.x. [p102]

G. Raskutti, M. J. Wainwright, and B. Yu. Lower bounds on minimax rates for nonparametric regression with additive sparsity and smoothness. In *Advances in Neural Information Processing Systems*, 2009. [p103]

G. Raskutti, M. J. Wainwright, and B. Yu. Minimax-optimal Rates for Sparse Additive Models over Kernel Classes via Convex Programming. *J. Mach. Learn. Res.*, 13(1):389–427, Feb. 2012. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2503308.2188398. [p103]

P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman. Sparse additive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(5):1009–1030, 2009. doi: 10.1111/j.1467-9868.2009.00718.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2009.00718.x. [p103]

O. Roustant, D. Ginsbourger, and Y. Deville. DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. *Journal of Statistical Software, Articles*, 51(1):1–55, 2012. ISSN 1548-7660. doi: 10.18637/jss.v051.i01. URL https://www.jstatsoft.org/v051/i01. [p104, 105, 116, 117, 118]

A. Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145(2):280 – 297, 2002. ISSN 0010-4655. doi: https://doi.org/10.1016/S0010-4655(02)00280-1. URL http://www.sciencedirect.com/science/article/pii/S0010465502002801. [p102]

A. Saltelli, S. Tarantola, and K. P.-S. Chan. A Quantitative Model-Independent Method for Global Sensitivity Analysis of Model Output. *Technometrics*, 41(1):39–56, 1999. doi: 10.1080/00401706.1999.10485594. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1999.10485594. [p104]

A. Saltelli, K. Chan, and E. Scott. *Sensitivity Analysis*. Wiley, 2009. ISBN 9780470743829. URL https://books.google.fr/books?id=gOcePwAACAAJ. [p110]

W. Schoutens. *Stochastic Processes and Orthogonal Polynomials*. Lecture Notes in Statistics. Springer New York, 2000. ISBN 9780387950150. URL https://books.google.fr/books?id=V2BS_Dmp0XoC. [p102]

I. M. Sobol. Sensitivity Estimates for Nonlinear Mathematical Models. In *Sensitivity Estimates for Nonlinear Mathematical Models*, 1993. [p101, 102]

C. Soize and R. Ghanem. Physical Systems with Random Uncertainties: Chaos Representations with Arbitrary Probability Measure. *SIAM Journal on Scientific Computing*, 26(2):395–410, 2004. doi: 10.1137/S1064827503424505. URL https://doi.org/10.1137/S1064827503424505. [p102]

B. Sudret. Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety*, 93(7):964 – 979, 2008. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2007.04.002. URL http://www.sciencedirect.com/science/article/pii/S0951832007001329. Bayesian Networks in Dependability. [p102]

A. W. Van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998. doi: 10.1017/CBO9780511802256. [p101]

W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris. Screening, Predicting, and Computer Experiments. *Technometrics*, 34(1):15–25, 1992. ISSN 00401706. URL http://www.jstor.org/stable/1269548. [p102]

N. Wiener. The Homogeneous Chaos. *American Journal of Mathematics*, 60(4):897–936, 1938. ISSN 00029327, 10806377. URL http://www.jstor.org/stable/2371268. [p102]

Y. Yang and H. Zou. A Fast Unified Algorithm for Solving Group-lasso Penalize Learning Problems. *Statistics and Computing*, 25(6):1129–1141, Nov. 2015. ISSN 0960-3174. doi: 10.1007/s11222-014-9498-5. URL http://dx.doi.org/10.1007/s11222-014-9498-5. [p109]

Y. Yang, H. Zou, and S. Bhatnagar. *gglasso: Group Lasso Penalized Learning Using a Unified BMD Algorithm*, 2020. URL https://CRAN.R-project.org/package=gglasso. R package version 1.5. [p109]

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006. doi: 10.1111/j.1467-9868.2005.00532.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2005.00532.x. [p106, 109]

Halaleh Kamari
UMR 8071 Laboratoire de Mathématiques et Modélisation d'Évry (LaMME)
Bâtiment IBGBI de l'Université d'Evry Val d'Essonne, 23 Bd de France, 91037 Evry CEDEX
France
halala.kamari@gmail.com

Sylvie Huet
UR 1404 Mathématiques et Informatique Appliquées du Génome à l'Environnement (MaIAGE)
Bâtiment 210 de INRAE, Domaine de Vilvert, 78352 JOUY-EN-JOSAS Cedex
France
shuet.inra@gmail.com

Marie-Luce Taupin
UMR 8071 Laboratoire de Mathématiques et Modélisation d'Évry (LaMME)
Bâtiment IBGBI de l'Université d'Evry Val d'Essonne, 23 Bd de France, 91037 Evry CEDEX
France
marie-luce.taupin@univ-evry.fr

Measuring the Extent and Patterns of Urban Shrinkage for Small Towns Using R

by Cristiana Vîlcea, Liliana Popescu and Alin Clincea

Abstract Urban shrinking is a phenomenon as common as urban expansion nowadays and it affects urban settlements of all sizes, especially from developed and industrialized countries in Europe, America and Asia. The paper aims to assess the patterns of shrinkage for small and medium sized towns in Oltenia region (Romania), considering demographic, economic and social indicators with a methodological approach which considers the use of different functions and applications of R packages. Thirteen selected indicators are analysed to perform the multivariate analysis on Principal Component Analysis using the `prcomp()` function and the **ggplot2** package to visualize the patterns of urban shrinkage. Two composite indicators were additionally created to measure the extent of urban shrinkage: CSI (Composite Shrinking Index) and RDC (Regional Demographic Change) for two-time intervals. Based on the CSI, three major categories of shrinking were observed: persistent shrinkage, mild shrinking or slow evolution toward shrinking, where the vast majority of towns are found (including mining towns, where there still is a delayed restructuring of state-owned enterprises, and towns characterised by the agrarization of local economies), and stagnant/stabilized shrinkage.

1 Introduction

Shrinking cities, considered up until recently a politically taboo subject in Europe, systematically disregarded as a dominant development trend of some urban areas (Wiechmann, 2007; Nelle et al., 2017), and a stigmatized topic in planning research (Pallagst, 2010), are ever more present among the research topic of various scholars throughout the world, as well as the agenda of public authorities and policy makers. Thus, the concept of shrinking cities differs from the classic notion of urban decline since new processes are at stake (Cunningham-Sabot et al., 2014).

From a historical perspective, the current period of shrinkage is distinguishable from the earlier period of decline and an earlier period of growth as well, mainly due to the prevalence of population loss, less so in its severity and not in its persistence or lack thereof (Beauregard, 2013). Today, the concept of shrinking cities connotes the urban degenerative effects of the breakdown of Fordist agglomeration economies, as well as the effects of urban agglomerative and dissipative forces associated with the global diffusion of contemporary, post-Fordist systems of production (Audirac, 2014).

Beginning with the 20th century, shrinking cities have developed continuously into a global phenomenon, being located mainly in Central Europe, the US, Japan and Eastern European transformation countries (Oswalt and Rieniets, 2006). They tend to have a common industrial past and are now faced with large challenges as a result of economic restructuring (Urban Audit, 2007). In Europe, the number of growing cities has been falling steadily since the 1960s, while almost a quarter (24%) of the cities have registered a medium-term decline (almost half of all the Russian, Polish and Romanian cities with a population over 200,000 inhabitants) and more than a third showed a clear downturn since 1990 (Turok and Mykhnenko, 2007). Still, the phenomenon is not equally spread throughout Europe, some countries and regions being more affected, i.e. post socialist countries, where almost half of cities can be described as shrinking (Wiechmann and Wolff, 2013), while others, such as France, experiencing a more-limited intensity in terms of number of cities affected and population loss (Wolff et al., 2013).

Since the publication of *Die Schrympfende Stadt (the Shrinking City)* (Göb, 1977) and *Coping with City Shrinkage* (Breckenfeld, 1978), some of the ways of understanding the city have changed in emphasis. Researchers' efforts have focused on capturing the main features of the phenomenon due to its role in the reconfiguration of the urban space, which resulted in a substantial literature on urban shrinking that is undoubtedly an incontrovertible and increasingly important phenomenon, and a major challenge for future urban policies (Agirre-Maskariano, 2019; Bernt et al., 2012; Mallach et al., 2017; Nelle et al., 2017; Pallagst, 2010; Wiechmann and Bontje, 2015; Wiechmann and Wolff, 2013).

The last two decades saw an emergence of a corpus of research and projects designed to assess general causes and models of shrinkage. Several authors identify four to five major drivers for shrinkage, which are often found in a combination of two or more of these causes: i) suburbanization (flight of people and jobs to the suburbs, hollowing out of the core city, triggered by urban sprawl)(Wiechmann, 2006; Wiechmann and Bontje, 2015; Cunningham-Sabot et al., 2014; Audirac, 2014; Reckien and Martinez-Fernandez, 2011; Wiechmann and Bontje, 2015), ii) economic decline and industrial transformation (Wiechmann, 2006; Rink et al., 2010; Haase et al., 2014), iii) demographic change (e.g. falling birth rates, outmigration in rural depopulation areas) (Rink et al., 2010; Haase et al., 2014; Wiechmann and Bontje, 2015); iv) structural upheaval (economic reorganization, collapse

of an entire political system, unrest, resettlements) and environmental pollution (Wiechmann, 2006; Haase et al., 2014; Cunningham-Sabot et al., 2014; Wiechmann and Bontje, 2015).

Over the past decades, shrinkage has become a "normal pathway" of development for cities and regions all across Europe (Rink et al., 2010). Recent writing and research on the shrinking cities have also focused on indicators of urban shrinkage, which should be necessarily dynamic and capable of detecting medium-term tendencies, distinguishing between episodic or acute shrinkage (when developments fluctuate in a certain period, but have an overall negative evaluation) and continuous or chronic shrinking (Wolff and Wiechmann, 2014). Since population loss, which is an initial clue of urban processes and a major indicator for urban shrinkage, does not encompass the various aspects of the phenomenon, there is a continuous focus on indicators to measure shrinking cities. They can be summarized into three main categories: demographic indicators, economic indicators and social indicators.

Among the demographic indicators, migration, population natural increase and a shifting population structure (ageing and feminization) are key factors of population decline, highly connected to other social, economic and built environment variables (Turok and Mykhnenko, 2007; Guimarães et al., 2015; Haase et al., 2016; Bănică et al., 2017; Cauchi-Duval et al., 2017; Hartt, 2018). Indicators related to economic changes (employment, unemployment, firms, services) are more difficult to gather and raise issues in terms of comparability among countries (Martinez-Fernandez et al., 2015), but should not be neglected as the economic decline is among the drivers of shrinkage. Social indicators usually considered when analysing shrinkage refer either to households (housing permit rates, housing start rate, number of households) (Bontje, 2004; Wolff and Wiechmann, 2014; Guimarães et al., 2015; Lauf et al., 2016; Hartt, 2018), or to the number of students enrolled in education units (Guimarães et al., 2015). This decreases attractiveness of a city as an educational place. Consequently, it causes a decreasing number of students enrolled in compulsory education levels and the closure of educational institutions (Wolff and Wiechmann, 2014).

Most of the studies regarding shrinking cities focus on larger urban centres, exceeding two hundred thousand inhabitants. However, many of the spatial planning and development strategies of the European Union focus on small and medium-sized towns, often seen as the chronic patients of regional policy, constantly in need of care but never getting well (Wirth et al., 2016). Most of the French shrinking cities are small urban areas, with less than 50,000 inhabitants, while in Germany the most dramatic decrease occurs in medium sized cities (Martinez-Fernandez et al., 2015). In Hungary, almost every small town has shrunk during at least the last decade (Pirisi et al., 2015). In Romania, the discussion within urban planning has also begun, despite focusing on some larger towns, researchers from various fields drawing attention to this issue (Bănică et al., 2017; Jucu et al., 2016; Jucu and Pavel, 2019; Păun Constantinescu, 2019; Popescu, 2014; Schoenberg and Constantin, 2014; Stoica et al., 2020a,b) since the Romanian towns clearly witness a decline on short or medium term, which is uneven, but in perpetuum (Păun Constantinescu, 2013).

2 Data and methods

The multidimensional concept of shrinking cities has a comprehensive meaning, as the drivers which determine this process are complex (demographic, economic, social). Therefore official statistics from the last three censuses (1992, 2002, 2011) and from 2018 were used to calculate indicators. We applied simple standard formulae in population geography regarding the demographic and economic phenomena such as population natural increase, feminization, migration, ageing, unemployment, i.e. k_1-k_6 and construct composite indices. All indices used in the analysis of this phenomenon were based on their importance and relationship to each other and to the concept itself. After a thorough documentation focusing on other studies referring to shrinking cities (Haase et al., 2016; Hartt, 2018; Mallach et al., 2017; Rink et al., 2010; Wiechmann and Bontje, 2015; Wiechmann and Wolff, 2013) we selected a series of demographic and socio-economic indicators that met the criteria of availability and relevance for the study. Availability and consistency of data for small Romanian towns represented a major issue, as there are missing time series and uneven statistics. Thus, in order to perform the calculation without errors caused by missing values, authors imputed these missing values with zero, as the frequency of missingness was not high and it was observed that zero imputing does not affect the results of the study. This major issue played an important role in the selection of the best indicators considered for the multivariate analysis and the aggregation method. The chosen indicators vary in terms of measurement unit, as they are expressed as absolute or computed data (Table 1), but we also weighted their strengths and weaknesses.

The work flow consists in several stages, the main step concentrating on the creation and use of the composite indicators (CSI and RDC) (Figure 1).

As defined by OECD (2008) in the Glossary of Statistical terms *"a composite indicator is obtained*

Selected indices	M.U.	Assigned code
Demographic indicators		
migration rate	‰	k_1
rate of natural increase	‰	k_2
feminization index	%	k_3
elderly population	%	k_4
mean age	%	k_5
Economic indicators		
unemployment rate	%	k_6
employees	number	k_7
Social indicators		
households	number	k_8
libraries	number	k_9
doctors	number	k_{10}
students	number	k_{11}
education	number	k_{12}
hospital beds	number	k_{13}

Table 1: The selection of indicators used for aggregation.

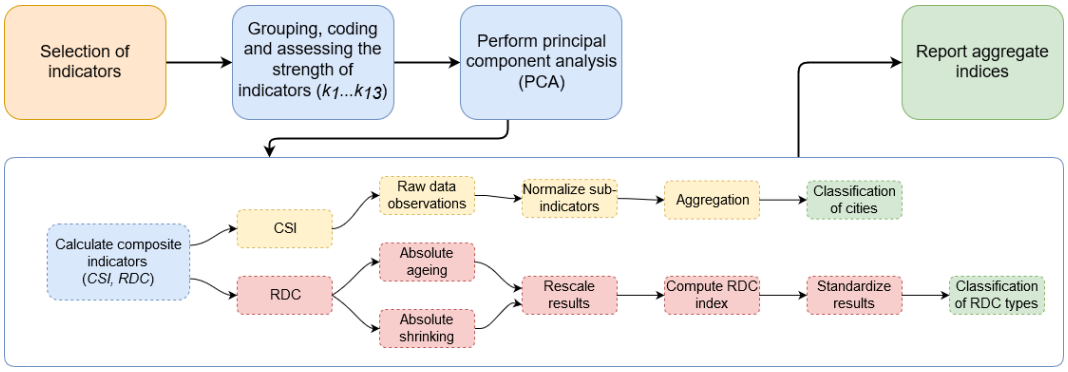


Figure 1: Overall workflow for the research methodology, depicting the main steps of the process, from the selection of indicators to the resulting aggregate indices.

from the combination of individual indicators into a single index, on the basis of an underlying model of the multidimensional concept that is being measured".

The study uses two types of composite indices to observe and analyse the dynamics of the shrinking process specific for the cities in the South West Development Region of Oltenia. The first composite indicator used is the Regional Demographic Change (RDC) obtained by applying the methodology described by Tivig et al. (2008) in the *Final Report of the Mapping Regional Demographic Change and Regional Demographic Location Risk in Europe*. The RDC index was initially used to describe the process of ageing and shrinking of the population in 27 European states from 264 regions. We adapted the methodology to compute this index for 34 towns located in the South West Development Region of Oltenia. The RDC index captures the extent and time-path of demographic change by accounting for two of its dimensions: population ageing with the perspective of shrinking. The RDC index is calculated using raw indicators for age, like the mean age, and population size like population density. Based on these two indicators for population age and density, the ageing and shrinking process are calculated and assessed in absolute terms for each town and three periods of time (please see research data, sheet *mean_age* and *density* for calculated data). Thus, the index was calculated for 3 time-intervals: 1992-2018; 2018-2030 and 1992-2030. In order to be able to make cross-period comparison, a rescaling was necessary. For analyzing the relative position of the towns within the region in terms of ageing and shrinking the values computed previously had been z-standardized (Tivig et al., 2008). If the RDC index shows the demographic change that takes place in the small towns under analysis, the RDC type indicate whether their population ages faster or slower and grows less or shrinks as compared to the average of the region. The four types of the RDC-index were applied to one regional dimension (South West Development Region of Oltenia) and represented showing the overall trend (1992-2030) and the future trend (2018-2030).

The second composite indicator (CSI) was created based on thirteen sub-indices ($k_1 \dots 13$) computed for all 34 towns located in the study area and aggregated into a single index using the additive method (Gan et al., 2017; Nardo et al., 2005; OECD, 2008; Saisana and Tarantola, 2002; Syrovátka and Schlossarek, 2019). The additive aggregation method, and especially the weighted arithmetic mean, is the widest used method (Gan et al., 2017; Pollesch and Dale, 2015) as it sums up the normalized values of sub-indicators. The Principal Component Analysis (PCA) was performed to analyse the correlation between the sub-indicators. We intended to use similar indicators as in other studies (Table 1), but approach a different method and process data in R. The multiple packages make this language a powerful tool for data processing and visualization (Barry, 2018) with multiple possibilities of application, especially in spatial analysis (Lovelace et al., 2019). As the sub-indicators had different measurement scales, variable standardization was handled using the option `scale=true` in `prcomp()`. Complying with the minimum required rule of 3:1 (Jollands, 2003), the use of this method allows summarizing the entire set of individual indicators (Paul et al., 2013; Jolliffe and Cadima, 2016), while preserving the maximum possible proportion of the total variation in the original data set (Sharma, 1996). The PCA results were represented using the **ggplot2** package (Kassambara, 2017; Wickham, 2016).

3 Results

PCA was used to analyse the interrelationships among all thirteen variables (Table 1) observed in 34 small towns located in Oltenia region, in two reference years 1992 and 2018. The main characteristic of this method is the reduction of a great number of variables to a smaller number (principal components), as linear combinations of the variables in the multivariate set, with minimum loss of information. Principal components outputs where computed using `prcomp()` for an improved numerical accuracy and displayed using `screeplot()`.

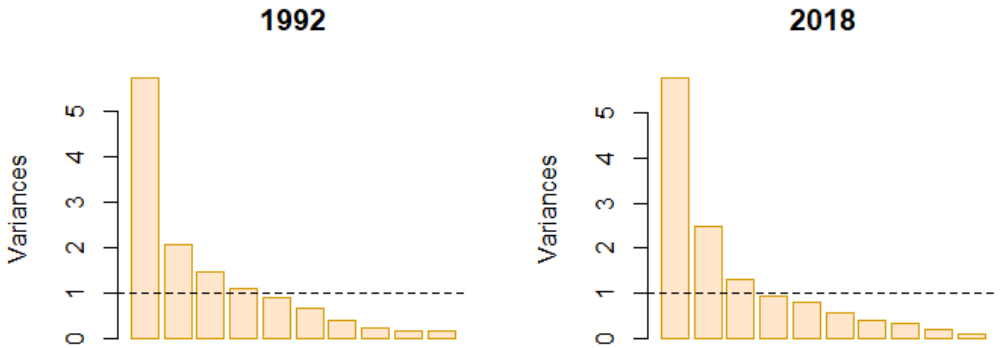


Figure 2: Scree plot of the principal components. The graphic representation of variances allows for easier selection of relevant components.

Analysing the scree plots, there are small differences in the eigenvalues of the components comparing the two years observed (Figure 2). If we use the Kaiser criterion, we see that only the first four (1992), respectively three (2018) components have eigenvalues greater than 1. But these components explain only 79.5% of the variation in the data for 1992 and 73.2% for 2018 and usually a cumulative proportion of at least 85%, that explain an acceptable level of variance, is obtained by the first five principal components. The fourth and fifth eigenvalues of the principal components in 2018 are only close to the value of 1. Therefore, values which are greater than 0.75 are considered as "strong". For an acceptable level of variance, we selected the first five principal components for analysis (Table 2).

Preliminary data analysis indicates a small difference in variance and proportion for the first five components, in the two reference years (Table 3). The data showed that the highest value of the variable is 0.690, respectively 0.634 (Table 4), therefore we considered that the level of correlation between each component is important above the value of 0.3.

For this set of results, the first principal component is strongly correlated with almost all the original variables of the social and economic factors in 1992 and 2018. It has exclusively negative correlations with almost all (except unemployment rate) variables observed in 2018. So, this component measures primarily the social-economic factors that influence the shrinking process of small cities. The second component measures the demographic dimension of shirking, as it has negative correlations for

1992	PC(1)	PC(2)	PC(3)	PC(4)	PC(5)	PC(6)	PC(7)	PC(8)	PC(9)	PC(10)	PC(11)	PC(12)	PC(13)
Std. dev.	2.40	1.48	1.16	1.04	0.95	0.83	0.63	0.47	0.40	0.37	0.29	0.21	0.15
Variance	0.44	0.17	0.10	0.08	0.07	0.05	0.03	0.02	0.01	0.01	0.01	0.00	0.00
Cumulative	44.2%	61.0%	71.3%	79.5%	86.5%	91.7%	94.8%	96.5%	97.8%	98.8%	99.5%	99.8%	100.0%
2018	PC(1)	PC(2)	PC(3)	PC(4)	PC(5)	PC(6)	PC(7)	PC(8)	PC(9)	PC(10)	PC(11)	PC(12)	PC(13)
Std. dev.	2.39	1.57	1.16	0.97	0.92	0.75	0.62	0.56	0.45	0.30	0.27	0.21	0.18
Variance	0.44	0.19	0.10	0.07	0.07	0.04	0.03	0.02	0.02	0.01	0.01	0.00	0.00
Cumulative	44.0%	62.9%	73.2%	80.4%	86.9%	91.3%	94.3%	96.6%	98.2%	98.8%	99.4%	99.7%	100.0%

Table 2: Summary of Principal Component Analysis (PCA) used for the selection of principal components. The components with a cumulative variance above 0.75 are considered "strong".

	PC1	PC2	PC3	PC4	PC5
1992	44.2%	61.0%	71.3%	79.5%	86.5%
2018	44.0%	62.9%	73.2%	80.4%	86.9%
change	-0.0018	0.0188	0.0190	0.0083	0.0042

Table 3: Changes in the variances of the first five principal components in the analyzed time interval: 1992-2018.

the rate of natural increase and positive correlations for the rest of the demographic variables like the feminization index, the percent of elders and the mean age. This fact suggests that the five criteria in 1992 and six in 2018 vary together, increasing or decreasing. The third and fourth components indicate correlations with both economic and demographic variables. The third component has negative correlations in 1992 and positive in 2018 for the migration rate and unemployment rate. In 2018, the third component is also strongly correlated with the number of employees. The fourth component shows a shift in correlations, as they are positive in 1992 with the introduction of a variable from the social sector (number of schools) and negative in 2018 for the demographic variables, but positive for the unemployment rate. The fifth component indicates also several changes in 2018 as compared to 1992, because in 2018 it has correlations only for the social variables (positive for the number of doctors and hospital beds and negative correlations for the number of libraries and schools) (Table 4).

The scope of this analysis is to identify the directions (principal components) along which the variation in the data is the highest. As the first two components account for most of the variance in the date, to achieve this goal, the dimensionality of the multivariate data was reduced into two principal components, in order to be able to visualize them graphically. The two graphs were generated in R using the **ggplot2**, **ggrepel**, **gridExtra** and **readxl** packages (see `shrinking_cities.r` from supplemental material).

The graphic visualization of the results allows to analyse the variance of the variables and the change in direction trends for 1992 and 2018. Analysing the scores of the second principal component versus the scores of the first principal component, we notice that, in 1992, the first principal component has large positive associations with the number of houses, libraries, doctors, pupils, schools and hospital beds and negative associations with the number of employees. So, it measures mainly the social dimension of the cities as shrinkage factors. The second component measures primarily the

	Principal Component (1992)					Principal Component (2018)				
Variable	PC(1)	PC(2)	PC(3)	PC(4)	PC(5)	PC(1)	PC(2)	PC(3)	PC(4)	PC(5)
k_1	0.076	-0.338	-0.644	0.064	-0.419	0.407	-0.035	-0.648	-0.476	-0.023
k_2	0.000	-0.588	-0.175	-0.713	-0.042	0.128	-0.613	-0.013	-0.614	-0.084
k_3	-0.350	0.642	-0.129	-0.522	0.278	-0.621	0.600	-0.153	-0.015	0.133
k_4	-0.711	0.557	0.089	-0.113	-0.240	0.338	0.843	-0.279	-0.047	-0.158
k_5	-0.483	0.636	0.215	0.128	-0.130	0.043	0.893	0.164	-0.154	-0.010
k_6	0.233	0.487	-0.708	0.142	-0.096	0.094	-0.384	-0.671	0.528	-0.060
k_7	0.903	-0.175	0.223	0.170	0.125	-0.757	-0.215	0.397	-0.021	0.186
k_8	0.935	0.212	0.051	0.064	-0.022	-0.936	-0.031	0.045	0.001	-0.210
k_9	0.683	0.266	0.145	-0.056	-0.546	-0.832	0.071	-0.183	-0.095	-0.342
k_{10}	0.806	0.355	-0.322	-0.054	0.186	-0.841	0.045	-0.286	-0.037	0.366
k_{11}	0.947	0.020	0.148	0.015	0.005	-0.900	-0.067	-0.083	-0.005	-0.211
k_{12}	0.697	0.175	0.313	-0.433	-0.323	-0.871	-0.030	-0.011	-0.018	-0.395
k_{13}	0.798	0.239	-0.140	-0.104	0.337	-0.783	0.004	-0.238	-0.116	0.517

Table 4: The first five values of the principal components for the variables k_1-k_{13} in 1992 and 2018.

demographic dimension as it has large positive loadings on the rate of elders and mean age (Figure 3, 1992). This aspect is changing entirely in 2018, if we observe the directions along which we have the highest variance of the date. Thus, the demographic variables (migration rate and elders) strongly influence component 1, while the social-economic variables influence the second component. For a better view of differences amongst towns, the points representing the urban settlements, vary in size and intensity according to the values of the third and fourth components (Figure 3, 2018).

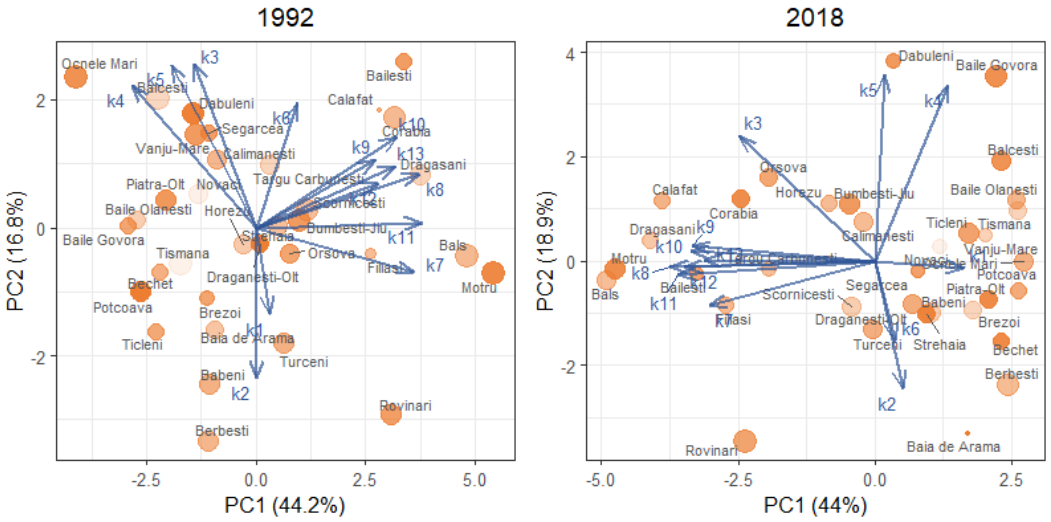


Figure 3: Principal Components Analysis of shrinkage indicators

The changes in population, especially the processes of ageing and shrinking, have a direct influence on the socio-economic dimension of settlements. As in most European countries, the ageing of the population is a phenomenon that characterizes almost all the settlements in Romania and will continue, but with different intensities; however, ageing greatly varies among the towns considered for analysis. The dimension of shrinking is given by the decrease in the number of inhabitants, which is better expressed by the decrease in the population density. The data indicated that, same as the ageing process, depopulation is another characteristic of the selected towns.

We used the RDC index to compare, at regional level, the dimension of ageing and shrinking of all 34 small towns located in Oltenia. Three time-intervals were considered (1992-2018 - past; 2018-2030 - future and 1992-2030 - overall) to show the overall and the future trend (Figure 4).

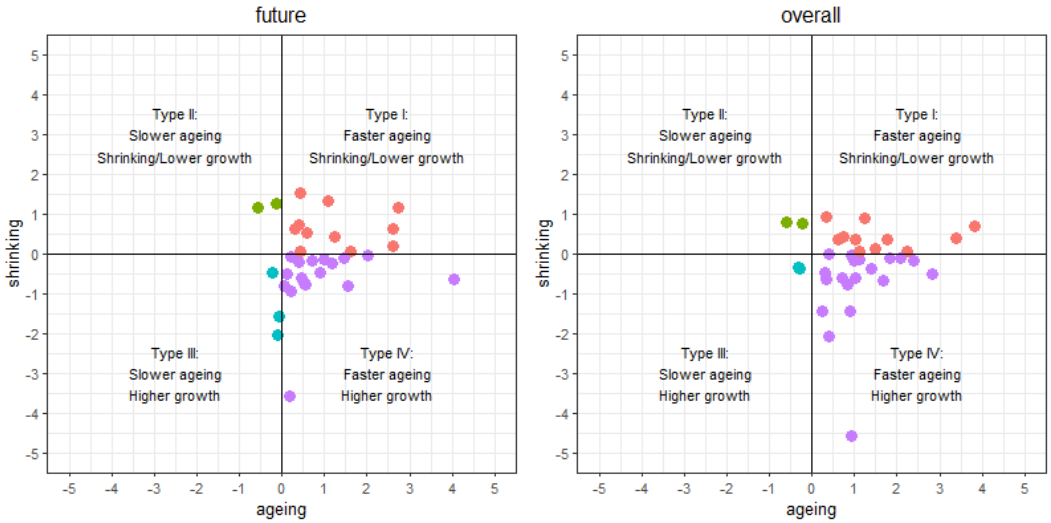


Figure 4: Future (2018-2030, left) and Overall (1992-2030, right) trends of Regional Demographic Change (RDC-type) for small towns in South-West Development Region of Oltenia

After examining the population trend for every town, we identified four categories, differentiated by ageing, population evolution and shrinking. This shows that some towns are more resilient to shrinkage than others, due to a combination of factors. However, there is clear evidence that a

significant number of towns (about 40%) are shrinking, while 80% face a fast-ageing process of the population.

The results indicated that there are some demographic changes between the future trend and the overall trend. If the share of towns included in the first two categories remains unchanged during the two analysed periods, their number shows a slight change for the fourth and third types. These results indicate that some settlements will pass in the future from higher growth and slower ageing (type III) to higher growth and faster ageing (IV) (Figure 4 & Table 5).

RDC type	Description	Future trend 2018-2030		Overall trend 1992-2030	
I	Faster ageing. Shrinking/Lower growth	11 towns	32%	11 towns	32%
II	Slower ageing. Shrinking/Lower growth	2 towns	6%	2 towns	6%
III	Slower ageing. Higher growth	4 towns	12%	2 towns	6%
IV	Faster ageing. Higher growth	17 towns	50%	19 towns	56%

Table 5: Percent of towns according to the four types of RDC

Using the composite indicator (CSI) we computed the values for the beginning and present period and analysed the dimension of change for all 34 towns at demographic and social-economic levels for the future period (2030). The indicator includes all thirteen sub-indices (Table 1) considered important in the shrinking process of cities. The values of CSI for 1992 and 2018 were calculated using the following formula:

$$CSI = \sum_{i=1}^{n} \frac{k_i - min\,(k_i)}{n\,(max\,(k_i) - min\,(k_i))} \tag{1}$$

where:

n:	subindicator count
k_i:	selected subindicator
$min\,(k_i)$:	minimum value of subindicator
$max\,(k_i)$:	maximum value of subindicator

To observe the shrinking trend of small towns for future period, the values for year 2030 were calculated based on the computed values of CSI for 1992, 2002, 2011 and 2018 using the forecast function in Microsoft Office Excel to predict the values for 2030. Standard deviation (St.dev.) values were calculated to observe highest values. The results were compared and represented, observing that thirteen towns have high variations and St.dev. values > 0.04 (Table 6 & Figure 5).

Towns	1992	2018	2030	St.dev.	Towns	1992	2018	2030	St.dev.
Băilești	0.599	0.554	0.529	0.036	Vânju-Mare	0.298	0.318	0.307	0.010
Bechet	0.235	0.189	0.202	0.023	Balș	0.551	0.593	0.629	0.039
Calafat	0.623	0.568	0.514	0.055	Corabia	0.555	0.486	0.482	0.041
Dăbuleni	0.297	0.400	0.446	0.076	Drăgănești-Olt	0.306	0.301	0.327	0.014
Filiași	0.559	0.491	0.431	0.064	Piatra-Olt	0.245	0.247	0.249	0.002
Segarcea	0.347	0.339	0.340	0.004	Potcoava	0.157	0.248	0.294	0.070
Tismana	0.342	0.291	0.238	0.052	Scornicești	0.418	0.365	0.410	0.028
Turceni	0.349	0.329	0.374	0.022	Băbeni	0.269	0.291	0.374	0.055
Bumbești-Jiu	0.373	0.326	0.338	0.025	Băile Govora	0.255	0.232	0.237	0.012
Novaci	0.393	0.414	0.422	0.015	Băile Olănești	0.277	0.245	0.263	0.016
Țicleni	0.259	0.225	0.220	0.021	Bălcești	0.350	0.236	0.214	0.073
Târgu Cărbunești	0.419	0.458	0.539	0.061	Berbești	0.196	0.196	0.222	0.015
Motru	0.495	0.531	0.572	0.039	Brezoi	0.296	0.281	0.302	0.011
Rovinari	0.356	0.365	0.427	0.038	Călimănești	0.321	0.385	0.447	0.063
Baia de Aramă	0.290	0.323	0.335	0.024	Drăgășani	0.572	0.630	0.709	0.069
Orșova	0.337	0.476	0.547	0.107	Horezu	0.341	0.441	0.485	0.074
Strehaia	0.361	0.267	0.236	0.065	Ocnele Mari	0.231	0.200	0.190	0.021

Table 6: The computed values of CSI for each town, showing shrinkage in 1992, 2018 and the 2030 forecast.

4 Discussions

As we initially calculated the RDC index for all 34 towns to analyse the dimension of shrinking, we deem necessary to use a more complex index to include economic and social dimensions. Tivig et al.

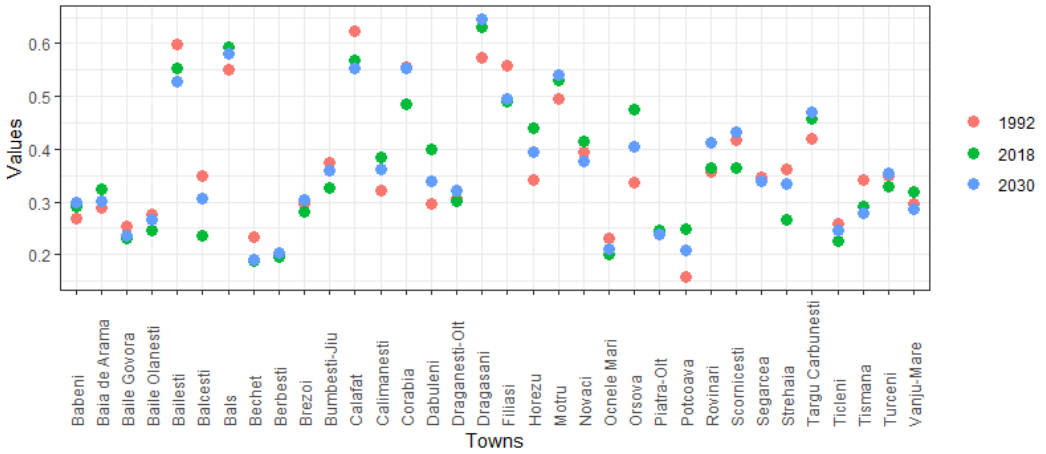


Figure 5: Variation of CSI values for the reference period

(2008) analyses the dimension of shrinking using only demographic indices, classifying cities in four categories. However, the scientific literature considers the topic of shrinking cities as a more complex process, which involves social and economic factors (Bernt et al., 2012; Martinez-Fernandez et al., 2015; Wolff and Wiechmann, 2014).

Besides the general factors, local features also contribute to the shrinking process of cities, especially for smaller ones. Therefore, we deem necessary to also use another, more complex, composite indicator. After comparing the results, we concluded that only six towns undergo a clear shrinking process, falling into the category of persistent shrinkage, while eighteen have a moderate or slow evolution toward shrinking (recent or mild shrinkage). There are also ten settlements that had a linear, almost stagnant, evolution for the entire reference period with the same future trend (Figure 6).

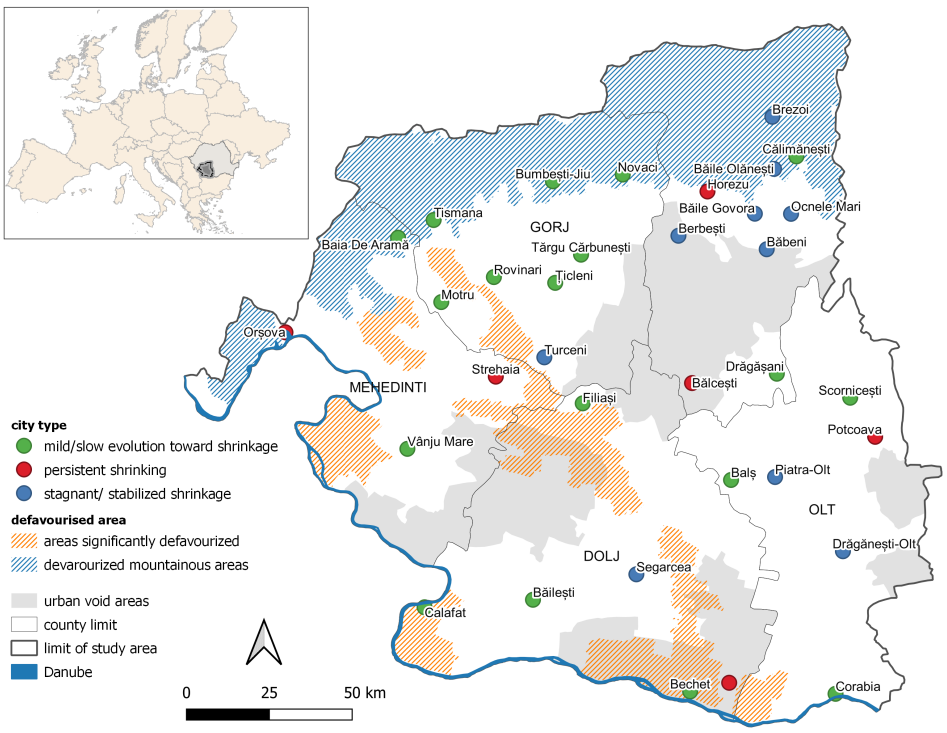


Figure 6: City types according to CSI

According to CSI values the towns can be classified in three categories: persistent shrinkage, a mild evolution towards shrinkage and stagnant shrinkage. The first category includes only 6 towns, which is less than a fifth of the analysed towns, and except for Orșova, they did not benefit during the communist period from any state-owned flagship companies and did not have any significant industrial base. This category varies largely in terms of population size, population loss, economic

activities and rural/urban characteristics. The smaller and more rural in character the town, the greater the pressure on shrinkage is. Just as in other regions, the regional and local economic restructuring alter the urban communities (Jucu et al., 2016).

A mild/slow evolution toward shrinkage characterizes the towns of Băilești, Bechet, Calafat, Tismana, Bumbești, Novaci, Țicleni, Târgu Cărbunești, Filiași, Motru, Rovinari, Baia de Aramă, Vânju Mare, Balș, Corabia, Scornicești, Călimănești and Drăgășani. There is a diversity of different town types (resourced-based mining towns, old ports, rural small towns, some tourism activities), scattered throughout the entire region, testifying that there are multiple factors causing shrinkage. Few of these towns have managed to keep their industrial activities due to the strategic importance of their power plants for the production of electricity. Most of them did not succeed to foster any new economic activities, nor modernise traditional branches, so there were no major new investments since the 1980s. This situation may be partially attributed to the towns location and poor accessibility, but also lack of entrepreneurial culture and poor administration that was hardly able to cope with the economic decline and transition to the market economy.

A stagnant/stabilized shrinkage was noted for ten towns (Figure 6). Some of these towns have kept and developed some forms of tourism activities (holiday homes, private camps, accommodation and leisure facilities), benefiting from considerable investments following various programmes financed by the EU for fostering the tourism sector.

Small towns that relied heavily on agriculture, forestry or mining activities will continue to be affected by shrinking, which will definitely put a stain on their physical and social structure, the need for viable alternatives to smart shrinkage being of utmost importance.

This analysis provides empirical confirmation that many small towns in Oltenia are in fact shrinking, despite different economic and demographic background, and that mainly the demographic indicators (population decrease, out-migration and ageing), followed by economic ones, are the most important factors for urban shrinkage in the region. However, these towns do not only face loses of population, but also a mismatch between the physical urban structure and the population needs.

What are the main causes for shrinking in the case of small and medium towns in Oltenia? After analysing the situation in other European countries, we can definitely say that the experience of this region has some specifics, as the urban shrinkage is not caused by suburbanization and to a larger extent, neither by deindustrialization. In 1992, most (3/4) of the small and medium-small towns in the region had a location quotient of industry below the urban average of 1.029 (Popescu, 2014), which might in part explain this situation. Moreover, these towns were ranked much lower in the urban hierarchy and had limited economic options, hosting mainly local resource-based industries.

Shrinking is rather caused by the overlapping of simultaneous demographic processes - out-migration, drops in birth rates, ageing (which leads to population decline) and economic difficulties (loss of jobs, lack of private companies, lack of entrepreneurial culture, heavy reliance on agriculture and subsidies from the government).

The current study provides a critical analysis on the extent of shrinking phenomenon that severely affects more than a third of the small and medium-size towns in the region of Oltenia and the outcomes of the post-socialist urban changes. In the long run, this shrinkage will definitely affect labour and the social and technical infrastructure, putting a great strain on the local budgets.

5 Conclusions

During the last decade, the issue of *shrinking cities* is pervasive for the research interest of academics and the political discourse of many countries. Unfortunately, despite the evident urban decline of small Romanian towns, from both the economic and demographic perspective during the last 3 decades, the politicians and authorities seem to miss the extent of this phenomenon, which has become rather common in Romania and Oltenia region as well.

The decrease in size of the population, the increase of the mean age and the ageing of the population has repercussions on the economy by reducing the labour force. It also influences the values of social criteria directly influencing the number of households, number of schools or the school population. The evolution of settlements may also correlate with the local factors like their proximity to a disfavoured area or to a dominantly rural area.

The thirteen indicators taken into consideration reflect demographic, economic and social aspects that point to vulnerabilities and lack of adaptability in Romanian shrinking cities (Bănică et al., 2017) rather than to aspects of resilience capacity. Also, the indicators were thought-out to be used as a base-model in analysing the dimension of a complex phenomenon like urban shrinking for other case studies which may include larger urban settlements. Of course, depending on other factors, the indicators considered for analysis may be modified. The three types of shrinkage testify for the

existence of different patterns (rhythms, trajectories, effects), which means that there is no miracle solution that could be applied to all the towns.

What are the alternatives for smart urban shrinkage? Considering that shrinking is seldom a rapid process, and it rather takes place over generations, providing the luxury of time to adapt the planning and designing process, the condition of shrinking does not have to be a terminal diagnosis for a small town (Fugate, 2007). Numerous studies point to the need for the involvement of communities to promote the principle of re-growing smaller and more sustainable. Another solution might focus on the assumed decision for turning to rural, considering the opportunity of using European funds and projects for rural development.

There are no policies for shrinking cities at national or regional level, despite the numerous strategies for sustainable development, which fail to acknowledge the extent and severity of this phenomenon, affecting mainly small and medium-size towns. All the strategies for development seem to disregard the context and consequences of urban shrinkage, choosing to focus instead mainly on economic growth, which seems to evade almost all these towns during the last decades. These towns lack any capacity to cope with the consequences of urban shrinkage, their development strategies lacking both awareness and preparedness to actively respond to this issue.

6 Author contribution

C.V. and L.P. conceived and wrote the content and the R code of the study. A.C. debugged and reviewed the R code; also prepared the LaTeX source.

Bibliography

M. Agirre-Maskariano. Politiques urbaines pour la mise en récit d'une ville moyenne périphérisée en décroissance : L'exemple de Montluçon. *Belgeo. Revue belge de géographie*, (3), 2019. ISSN 1377-2368. doi: 10.4000/belgeo.35087. URL http://journals.openedition.org/belgeo/35087. Number: 3 Publisher: National Committee of Geography of Belgium / Société Royale Belge de Géographie. [p123]

I. Audirac. Shrinking cities in the fourth urban revolution. In *Shrinking cities: international perspectives and policy implications*. Routledge, 2014. [p123]

T. Barry. Collections in R: Review and proposal. *The R Journal*, 10(1):455, 2018. ISSN 2073-4859. doi: 10.32614/RJ-2018-037. URL https://journal.r-project.org/archive/2018/RJ-2018-037/index.html. [p126]

R. Beauregard, A. Shrinking cities in the United States in historical perspective: A research note. In *Shrinking cities: International perspectives and policy implications*, volume 33, pages 49–57. Routledge, 2013. URL https://books.google.com/books/about/Shrinking_Cities.html?hl=ro&id=NWhtAAAAQBAJ. [p123]

M. Bernt, M. Cocks, C. Couch, K. Großmann, A. Haase, and D. Rink. Policy response, governance and future directions, 2012. URL https://www.ufz.de/export/data/400/39031_ResearchBrief2_.pdf. [p123, 130]

M. Bontje. Facing the challenge of shrinking cities in East Germany: The case of Leipzig. *GeoJournal*, 61: 13–21, 2004. URL https://link.springer.com/content/pdf/10.1007/s10708-005-0843-2.pdf. [p124]

G. Breckenfeld. Coping with city shrinkage. *Civil Engineering—ASCE*, 48(11):112–113, 1978. URL https://cedb.asce.org/CEDBsearch/record.jsp?dockey=0027669. Publisher: ASCE. [p123]

A. Bănică, M. Istrate, and I. Muntele. Challenges for the resilience capacity of Romanian shrinking cities. *Sustainability*, 9(12):2289, 2017. doi: 10.3390/su9122289. URL https://www.mdpi.com/2071-1050/9/12/2289. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute. [p124, 131]

N. Cauchi-Duval, F. Cornuau, and M. Rudolph. La décroissance urbaine en France : les effets cumulatifs du déclin. *Metropolitique*, page 11, 2017. URL https://www.metropolitiques.eu/IMG/pdf/met-cauchiduval-cornuau-rudolph.pdf. [p124]

E. Cunningham-Sabot, I. Audirac, S. Fol, and C. Martinez-Fernandez. Theoretical approaches of "shrinking cities". In *Shrinking Cities: International Perspectives and Policy Implications*, pages 30–46. Routledge, 2014. [p123, 124]

J. N. Fugate. *Shrinking gracefully: looking for effective planning and design approaches for small town America*. Thesis, Massachusetts Institute of Technology, 2007. URL https://dspace.mit.edu/handle/1721.1/39846. Accepted: 2008-01-10T14:25:30Z Journal Abbreviation: Looking for effective planning and design approaches for small town America. [p132]

X. Gan, I. C. Fernandez, J. Guo, M. Wilson, Y. Zhao, B. Zhou, and J. Wu. When to use what: Methods for weighting and aggregating sustainability indicators. *Ecological Indicators*, 81:491–502, 2017. ISSN 1470-160X. doi: 10.1016/j.ecolind.2017.05.068. URL http://www.sciencedirect.com/science/article/pii/S1470160X17303357. [p126]

M. H. Guimarães, A. P. Barreira, and T. Panagopoulos. Shrinking cities in Portugal – where and why. *Revista Portuguesa de Estudos Regionais*, 40:23–41, 2015. URL https://www.redalyc.org/pdf/5143/514351600002.pdf. [p124]

R. Göb. Die schrumpfende Stadt. *Archiv für Kommunalwissenschaften*, 16:149–177, 1977. [p123]

A. Haase, D. Rink, K. Grossmann, M. Bernt, and V. Mykhnenko. Conceptualizing urban shrinkage. *Environment and Planning A: Economy and Space*, 46(7), 2014. doi: 10.1068/a46269. URL https://journals.sagepub.com/doi/abs/10.1068/a46269. Publisher: SAGE PublicationsSage UK: London, England. [p123, 124]

A. Haase, M. Bernt, K. Großmann, V. Mykhnenko, and D. Rink. Varieties of shrinkage in European cities. *European Urban and Regional Studies*, 23(1):86–102, 2016. doi: 10.1177/0969776413481985. URL https://journals.sagepub.com/doi/abs/10.1177/0969776413481985. Publisher: SAGE PublicationsSage UK: London, England. [p124]

M. D. Hartt. How cities shrink: Complex pathways to population decline. *Cities*, 75:38–49, 2018. ISSN 02642751. doi: 10.1016/j.cities.2016.12.005. URL https://linkinghub.elsevier.com/retrieve/pii/S0264275116302943. [p124]

N. Jollands. *The usefulness of aggregate indicators in policy making and evaluation: a discussion with application to eco-efficiency indicators in New Zealand*. phdthesis, The Australian National University, 2003. Accepted: 2003-11-04 Last Modified: 2019-01-22 Publisher: The Australian National University. [p126]

I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065): 20150202, 2016. doi: 10.1098/rsta.2015.0202. URL https://royalsocietypublishing.org/doi/full/10.1098/rsta.2015.0202. Publisher: Royal Society. [p126]

I. S. Jucu and S. Pavel. Post-communist urban ecologies of Romanian medium-sized towns. *Forum geografic*, XVIII(2):170–183, 2019. doi: 10.5775/fg.2019.073.d. URL http://forumgeografic.ro/ro/2019/2577/. [p124]

I. S. Jucu, R. Creţan, and F. Timofte. Economic restructuring and contrasting urban patterns in Romanian post-socialist municipalities: the experiences of Timiş county, Romania. *Review of Historical Geography and Toponomastics*, XI(21):79–96, 2016. URL https://geografie.uvt.ro/wp-content/uploads/2015/06/11_JUCU_CRETAN_FABIAN.pdf. [p124, 131]

A. Kassambara. *R Graphics Essentials for Great Data Visualization*. CreateSpace Independent Publishing Platform, 1 edition, 2017. [p126]

S. Lauf, D. Haase, and B. KleinschmitcaTechnische. The effects of growth, shrinkage, population aging and preference shifts on urban development — a spatial scenario analysis of Berlin, Germany. *Land Use Policy*, 52:240–254, 2016. ISSN 0264-8377. doi: 10.1016/j.landusepol.2015.12.017. URL https://www.sciencedirect.com/science/article/abs/pii/S0264837715004135. Publisher: Pergamon. [p124]

R. Lovelace, J. Nowosad, and J. Muenchow. *Geocomputation with R*. The R Series. Chapman & Hall/CRC, 2019. [p126]

A. Mallach, A. Haase, and K. Hattori. The shrinking city in comparative perspective: Contrasting dynamics and responses to urban shrinkage. *Cities*, 69:102–108, 2017. ISSN 02642751. doi: 10.1016/j.cities.2016.09.008. URL https://linkinghub.elsevier.com/retrieve/pii/S0264275116303110. [p123, 124]

C. Martinez-Fernandez, T. Weyman, S. Fol, I. Audirac, E. Cunningham-Sabot, T. Wiechmann, and H. Yahagi. Shrinking cities in Australia, Japan, Europe and the USA: From a global process to local policy responses. *Progress in Planning*, 2015. doi: 10.1016/j.progress.2014.10.001. [p124, 130]

M. Nardo, M. Saisana, A. Saltelli, and S. Tarantola. Tools for composite indicators building, 2005. [p126]

A. Nelle, K. Großmann, D. Haase, S. Kabisch, D. Rink, and M. Wolff. Urban shrinkage in Germany: An entangled web of conditions, debates and policies. *Cities*, 69:116–123, 2017. ISSN 02642751. doi: 10.1016/j.cities.2017.02.006. URL https://linkinghub.elsevier.com/retrieve/pii/S026427511630227X. [p123]

OECD. *Handbook on Constructing Composite Indicators: Methodology and User Guide*. OECD Publishing, 2008. ISBN 978-92-64-04346-6. [p124, 126]

P. Oswalt and T. Rieniets. *Atlas of shrinking cities*. Hatje Cantz, 2006. ISBN 978-3-7757-1714-4. URL https://www.research-collection.ethz.ch/handle/20.500.11850/25547. Accepted: 2018-03-29T10:36:31Z Journal Abbreviation: Atlas der schrumpfenden Städte. [p123]

K. Pallagst. Viewpoint: The planning research agenda: shrinking cities — a challenge for planning cultures. *The Town Planning Review*, 81(5):i–vi, 2010. doi: 10.3828/tpr.2010.22. URL https://www.jstor.org/stable/27975967. [p123]

L. C. Paul, L. Saha, A. A. Suman, and N. U. Mondal. Methodological Analysis of Principal Component Analysis Method. *International Journal of Scientific & Engineering Research*, 4(3):9, 2013. [p126]

G. Pirisi, A. Trócsányi, and B. Makkai. Between shrinking and blooming: the crossroad of small towns' urbanisation in Hungary. *Annales Universitatis Paedagogicae Cracoviensis Studia Geographica*, 8 Small Towns' Development Problems:12–28, 2015. ISSN 2084-5456. URL http://cejsh.icm.edu.pl/cejsh/element/bwmeta1.element.desklight-6a8a7af9-e7ee-4f2d-a69a-3c35dbeda870. Publisher: Instytut Geografii Uniwersytetu Pedagogicznego im. Komisji Edukacji Narodowej w Krakowie. [p124]

N. Pollesch and V. H. Dale. Applications of aggregation theory to sustainability assessment. *Ecological Economics*, 114:117–127, 2015. ISSN 0921-8009. doi: 10.1016/j.ecolecon.2015.03.011. URL http://www.sciencedirect.com/science/article/pii/S0921800915000968. [p126]

C. Popescu. Deindustrialization and urban shrinkage in Romania. what lessons for the spatial policy? *Transylvanian Review of Administrative Sciences*, 10(42):181–202, 2014. ISSN 1842-2845. URL https://www.rtsa.ro/tras/index.php/tras/article/view/97. Number: 42. [p124, 131]

I. Păun Constantinescu. *Shrinking cities în România. O abordare contemporană a contracției și delinului urban*. phdthesis, Universitatea de Arhitectură și Urbanism Ion Mincu, 2013. URL https://www.uauim.ro/f/doctorat/sustineri_teze/REZUMAT.pdf. [p124]

I. Păun Constantinescu. *Shrinking Cities in Romania*. DOM Publishers, 2019 edition, 2019. ISBN 978-3-86922-372-8. URL https://dom-publishers.com/products/shrinking-cities-in-romania. [p124]

D. Reckien and C. Martinez-Fernandez. Why do cities shrink? *European Planning Studies*, 19(8):1375–1397, 2011. ISSN 0965-4313. doi: 10.1080/09654313.2011.593333. URL https://www.tandfonline.com/doi/abs/10.1080/09654313.2011.593333. Publisher: Routledge. [p123]

D. Rink, A. Haase, M. Bernt, and K. Großmann. Addressing urban shrinkage across Europe – challenges and prospects, 2010. URL https://www.ufz.de/export/data/400/39003_D9_Research_Brief_FINAL14223.pdf. [p123, 124]

M. Saisana and S. Tarantola. State of the art. Report on current methodologies and practices for composite indicator development, 2002. [p126]

A. M. Schoenberg and D. L. Constantin. Urban shrinkage in Romania: Scope and determinants. In *Shrinking Cities*. Routledge, 1 edition, 2014. ISBN 978-0-203-07976-8. Num Pages: 14. [p124]

S. Sharma. *Applied Multivariate Techniques*. Wiley, 1 edition, 1996. ISBN 978-0-471-31064-8. [p126]

I.-V. Stoica, A. F. Tulla, D. Zamfir, and A.-I. Petrișor. Exploring the Urban Strength of Small Towns in Romania. *Social Indicators Research*, 152(3):843–875, Dec. 2020a. ISSN 0303-8300, 1573-0921. doi: 10.1007/s11205-020-02465-x. URL http://link.springer.com/10.1007/s11205-020-02465-x. [p124]

I.-V. Stoica, D. Zamfir, and L. C. Săftoiu. Small towns in Romania: the road from the urban dream to the rural reality. In A. Steinführer, A.-B. Heindl, U. Grabski-Kieron, and A. Reichert-Schick, editors, *New Rural Geographies in Europe: Actors, Processes, Policies*. LIT Verlag, 2020b. ISBN 978-3-643-91302-9. Google-Books-ID: YqAeEAAAQBAJ. [p124]

M. Syrovátka and M. Schlossarek. Measuring development with inequality: How (should) aggregate indicators of development account for inequality? *Ecological Economics*, 164:106320, 2019. ISSN 09218009. doi: 10.1016/j.ecolecon.2019.04.032. URL https://linkinghub.elsevier.com/retrieve/pii/S0921800918307274. [p126]

T. Tivig, K. Frosch, and S. Kühntopf. Mapping regional demographic change and regional demographic location risk in Europe laboratory demographic change, 2008. [p125, 129]

I. Turok and V. Mykhnenko. The trajectories of European cities, 1960–2005. *Cities*, 24(3):165–182, 2007. ISSN 0264-2751. doi: 10.1016/j.cities.2007.01.007. URL https://www.sciencedirect.com/science/article/abs/pii/S0264275107000212. Publisher: Pergamon. [p123, 124]

Urban Audit. State of European cities report. Adding value to the European Urban Audit, 2007. URL https://ec.europa.eu/regional_policy/sources/docgener/studies/pdf/urban/stateofcities_2007.pdf. [p123]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p126]

T. Wiechmann. Types of shrinking cities–Introductive Notes on a Global Issue. In *International Symposium "Coping with City Shrinkage and Demographic Change–Lessons from around the Globe"*, Dresden, 2006. [p123, 124]

T. Wiechmann. What are the problems of shrinking cities? lessons learned from an international comparison. *The Future of Shrinking Cities-Problems, Patterns and Strategies of Urban Transformation in a Global Context*, pages 5–16, 2007. URL https://www.nature.com/articles/nmeth.4346. [p123]

T. Wiechmann and M. Bontje. Responding to tough times: Policy and planning strategies in shrinking cities. *European Planning Studies*, 23(1):1–11, 2015. ISSN 0965-4313. doi: 10.1080/09654313.2013.820077. URL https://www.tandfonline.com/doi/abs/10.1080/09654313.2013.820077. Publisher: Routledge. [p123, 124]

T. Wiechmann and M. Wolff. Urban shrinkage in a spatial perspective – operationalization of shrinking cities in Europe 1990 - 2010. *AESOP - ACSP Joint Congress*, page 21, 2013. [p123, 124]

P. Wirth, V. Elis, B. Müller, and K. Yamamoto. Peripheralisation of small towns in Germany and Japan – dealing with economic decline and population loss. *Journal of Rural Studies*, 47:62–75, 2016. ISSN 0743-0167. doi: 10.1016/j.jrurstud.2016.07.021. URL https://www.sciencedirect.com/science/article/abs/pii/S0743016716301851. Publisher: Pergamon. [p124]

M. Wolff and T. Wiechmann. Indicators to measure shrinking cities. In C. Martinez-Fernandez, S. Fol, and T. Weyman, editors, *A Conceptual Framework for Shrinking Cities*, Cost Action TU0803: Cities Regrowing Smaller (CIRES). Researchgate, 2014. URL https://www.researchgate.net/publication/270339860_Indicators_to_measure_shrinking_cities. [p124, 130]

M. Wolff, S. Fol, H. Roth, and E. Cunningham-Sabot. Shrinking cities, villes en décroissance : une mesure du phénomène en France. *Cybergeo : European Journal of Geography*, 2013. ISSN 1278-3366. doi: 10.4000/cybergeo.26136. URL http://journals.openedition.org/cybergeo/26136. Publisher: CNRS-UMR Géographie-cités 8504. [p123]

Cristiana Vîlcea
University of Craiova
Faculty of Sciences
Department of Geography
13 Al. I. Cuza Street, Craiova
Romania
cristiana.vilcea@gmail.com

Liliana Popescu
University of Craiova
Faculty of Sciences
Department of Geography
13 Al. I. Cuza Street, Craiova
Romania
popescu.liliana.ucv@gmail.com
corresponding author

Alin Clincea
University of Craiova
Faculty of Sciences
Department of Informatics
13 Al. I. Cuza Street, Craiova
Romania
alin.clincea@gmail.com

blindrecalc - An R Package for Blinded Sample Size Recalculation

by Lukas Baumann, Maximilian Pilz, and Meinhard Kieser

Abstract Besides the type 1 and type 2 error rate and the clinically relevant effect size, the sample size of a clinical trial depends on so-called nuisance parameters for which the concrete values are usually unknown when a clinical trial is planned. When the uncertainty about the magnitude of these parameters is high, an internal pilot study design with a blinded sample size recalculation can be used to achieve the target power even when the initially assumed value for the nuisance parameter is wrong. In this paper, we present the R-package **blindrecalc** that helps with planning a clinical trial with such a design by computing the operating characteristics and the distribution of the total sample size under different true values of the nuisance parameter. We implemented methods for continuous and binary outcomes in the superiority and the non-inferiority setting.

1 Introduction

Determining the sample size that is necessary to achieve a certain target power is a fundamental step in the planning phase of every clinical trial. The sample size depends on the type 1 error rate α, the type 2 error rate β, the effect size Δ and so-called nuisance parameters, which are parameters that affect the distribution of the test statistic but are not of interest in the test problem. While the type 1 and type 2 error rates are usually predetermined and the minimal clinically important effect size is known, there is often uncertainty about the magnitude of the nuisance parameters, such as the variance of the data σ^2 for tests with continuous outcomes or the overall response rate p for tests with binary outcomes.

Consider, as an example, the meta analysis by Nakata et al. (2018) that compares minimally invasive preservation with splenectomy during distal pancreatectomy. Among others, the overall morbidity of the two groups is compared and data from 13 studies are reported (cf. Figure 2(c) in Nakata et al. (2018)). Within these 13 studies, overall morbidity rates pooled over both groups between 0.10 and 0.62 are reported. This illustrates the high uncertainty about the "true" overall morbidity rate, which is the nuisance parameter in this setting.

In these cases, an internal pilot study design with blinded sample size recalculation can be used. In such a design, the nuisance parameter is estimated in a blinded way (i.e., without using information about the group assignment of the patients) after a certain number of outcome data is available, and the sample size is recalculated using this information (Wittes and Brittain, 1990). While in principle blinded sample size recalculation could be done without any a priori sample size calculation, it is still advisable to calculate an initial sample size based on the best guess for the nuisance parameter available in the planning phase and to determine when to recalculate the sample size based on this initial calculation. This is done to avoid conducting the recalculation too early (so that there is still a great uncertainty about the magnitude of the nuisance parameter when recalculation is performed), or too late (so that there may be no room for adjusting the sample size any longer as the recalculated sample size is already exceeded). Using this method to recalculate the sample size is an attractive option because the cost in terms of additional sample size is very small (depending on the outcome) and in most scenarios the type 1 error rate is unaffected by the blinded sample size recalculation. Hence, whenever there is uncertainty about the value of a nuisance parameter and the logistics of the trial allow it, blinded sample size recalculation can be used. Meanwhile, this is even recommended by regulatory authorities. For instance, the Committee for Medical Products for Human Use (CHMP) (2006) states that "(w)henever possible, methods for blinded sample size reassessment (...) should be used".

Methods to reassess the sample size in a blinded manner in an internal pilot study design have been developed for a variety of outcomes. Based on the early work by Stein (1945), Wittes and Brittain (1990) introduced the internal pilot study design for continuous outcomes. Their work was extended in different manners by different authors (cf. among others Birkett and Day (1994), Denne and Jennison (1999), and Kieser and Friede (2000)). In all these papers, the main task is to re-estimate the variance of a continuous outcome in a blinded way. These ideas can be applied to binary outcomes as well where the re-estimated nuisance parameter is the overall response rate over both treatment arms. Associated methods were, for instance, presented by Gould (1992) and Friede and Kieser (2004) for superiority trials and by Friede et al. (2007) for non-inferiority trials.

However, despite the clear benefit of a blinded sample size recalculation and a great number of publications on that topic, **blindrecalc** is to the knowledge of the authors the first R-package on

CRAN, and thus a freely available software, that helps with the planning of a clinical trial with such a design by computing the operating characteristics and the distribution of the total sample size of the study. The package can be used for pre-planned and midcourse implemented blinded sample size reassessments in order to evaluate the potential scenarios the blinded sample size re-estimation may imply. For continuous outcomes, we implemented the t-test for superiority trials and the shifted t-test for non-inferiority trials. For binary outcomes, we implemented the chi-squared test for superiority trials and the Farrington-Manning test for non-inferiority trials.

The structure of the paper is as follows: In the Statistical methods section, we explain the general way of proceeding when conducting a trial with an internal pilot study and how to obtain a blinded estimate of the nuisance parameter for continuous and binary outcomes. The structure of the package is introduced in Package structure. We demonstrate how **blindrecalc** can be utilized to plan a trial with an internal pilot study design and blinded sample size recalculation in Usage and example. In Development principles, we outline the principles of the development process and how we ensure the quality of our code. Finally, a brief Conclusion complements this paper.

2 Statistical methods

The general procedure for planning and conducting a trial with a blinded sample size recalculation is as follows: At first, an initial sample size n_{init} is calculated by using a best guess for the value of the nuisance parameter. The sample size for the first stage of the trial, n_1, is then calculated as a fraction of n_{init}, e.g., 0.25, 0.5 or 0.75. After n_1 observations are available, the total sample size n_{rec} is recalculated in a blinded way based on the available data. The final total sample size n is then determined as:

$$n = \min\left(\max\left(n_1, n_{rec}\right), n_{max}\right),$$

where n_{max} is a prespecified maximal sample size. This is called the unrestricted design with upper boundary. A restricted design would use n_{init} as a lower boundary (Wittes and Brittain, 1990). Often n_{max} is set to a multiple of n_{init}. The special case of $n_{max} = \infty$ results in the unrestricted design (Birkett and Day, 1994). After the final total sample size is calculated, the $n_2 = n - n_1$ observations for the second stage are gathered. Finally, the specified statistical test can be conducted with the data of all n patients.

In the following, we shortly introduce the implemented tests and how to obtain a blinded estimate of the nuisance parameter in each case.

Continuous outcomes

Assume a clinical two-arm trial with normally distributed outcomes where a higher value is deemed to be favorable, with mean values μ_E (experimental group) and μ_C (control group) and common unknown variance σ^2. The outcome of interest is the mean difference $\Delta := \mu_E - \mu_C$. By introducing a non-inferiority margin $\delta > 0$, the test problem is given by

$$H_0 : \Delta \leq -\delta \text{ vs. } H_1 : \Delta > -\delta.$$

The null hypotheses can be tested by a shifted t-test taking the non-inferiority margin δ into account. Note that the special case $\delta = 0$ corresponds to the standard t-test for superiority. The approximate total sample size for a one-sided t-test to detect a mean difference of $\Delta = \Delta^* > -\delta$ with a power of $1 - \beta$ while controlling the type 1 error rate at level α equals

$$n = n_E + n_C = \frac{(1+r)^2}{r} \frac{\left(z_{1-\alpha/2} + z_{1-\beta}\right)^2 \sigma^2}{\left(\Delta^* + \delta\right)^2}.$$

Here, r refers to the allocation ratio of the sample sizes between the experimental and the control group, i.e., $r = n_E/n_C$, and z_{1-q} denotes the $1-q$ quantile of the standard normal distribution.

In this framework, the nuisance parameter is the unknown variance σ^2. Due to potential uncertainty on the value of σ^2, it seems appropriate to re-estimate it in a blinded interim analysis to ensure that the desired power level is met. Inserting $\widehat{\sigma}^2$ observed mid-course into the above sample size formula may lead to a more reasonable sample size for the respective trial than sticking to the value assumed in the planning stage. There exist different methods for estimating σ^2 in a blinded manner

(Zucker et al., 1999). In **blindrecalc**, the one-sample variance estimator is implemented. It is defined as

$$\widehat{\sigma}^2 := \frac{1}{n_1 - 1} \sum_{j \in \{E,C\}} \sum_{k=1}^{n_{1,j}} \left(x_{j,k} - \bar{x} \right)^2,$$

where $x_{j,k}$ is the outcome of patient k in group j, $n_{1,j}$ denotes the first-stage sample size in group j, i.e., $n_1 = n_{1,E} + n_{1,C}$, and $\bar{x}$ equals the mean over all n_1 observations. Since the patient's group allocation is not considered when computing $\widehat{\sigma}^2$, blinding is maintained when using this variance estimator.

In the superiority case, i.e., if $\delta = 0$, blinded sample size reassessment can be performed without relevant type 1 error rate inflation (Kieser and Friede, 2003). In the non-inferiority case, however, the type 1 error rate may be inflated by the internal pilot study design and a correction of the applied significance level may become necessary to protect the nominal type 1 error rate at level α (Friede and Kieser, 2003). In particular, this inflation arises if the sample size recalculation is performed too early, i.e., if n_1 is chosen too small.

Interestingly, the cost of this procedure in terms of sample size is quite low. Since the one-sample variance estimate slightly overestimates the variance, an increase in sample size arises. However, this increase amounts to only 8 patients with $\alpha = 0.025$ and $\beta = 0.2$ or 12 patients with the same significance level and $\beta = 0.1$ (Friede and Kieser, 2001). In return, the sample size recalculation procedure implies that the trial's power meets the target value $1 - \beta$ for a wide range of values of σ^2.

Lu (2016) gives closed formulas for the exact distribution of the test statistic of the two-sample t-test in this setting. This allows the simulation of error probabilities and of the sample size distribution in an acceptable amount of time. The proposals made by Lu (2016) are implemented in **blindrecalc**. Thus, the design characteristics for continuous outcomes presented in **blindrecalc** are obtained by simulation and not by exact computation. This is the case for for binary outcomes that are presented in the following.

Binary outcomes

In a superiority trial with binary outcomes where a higher response probability is assumed to be favorable, the one-sided null and alternative hypothesis are

$$H_0 : p_E \leq p_C \text{ vs. } H_1 : p_E > p_C,$$

where p_E and p_C denote the event probabilities in the experimental and the control group, respectively. While several tests exist for this test problem, the widely used chi-squared test is implemented in **blindrecalc**. The sample size for this test can be approximated with the formula (Kieser, 2020):

$$n = \frac{1+r}{r} \frac{\left(z_{1-\alpha/2} \sqrt{(1+r) \cdot p_0 \cdot (1-p_0)} + z_{1-\beta} \sqrt{r \cdot p_{C,A} \cdot (1-p_{C,A}) + p_{E,A} \cdot (1-p_{E,A})} \right)^2}{\Delta^2}.$$

Again, r denotes to the allocation ratio of the sample sizes, and $z_{1-\alpha/2}$ and $z_{1-\beta}$ are the $1-\alpha/2$ and the $1-\beta$ quantiles of the standard normal distribution. Furthermore, $p_{C,A}$ and $p_{E,A}$ are the response probabilities in the control and the experimental group under the assumed alternative, p_0 is the overall response probability, i.e., $p_0 = (p_{C,A} + r \cdot p_{E,A}) / (1+r)$, and Δ is the effect under the alternative, i.e. $\Delta = p_{E,A} - p_{C,A}$. The nuisance parameter here is p_0, which can be estimated in a blinded way after n_1 observations with

$$\hat{p}_0 = \frac{X_{1,E} + X_{1,C}}{n_{1,E} + n_{1,C}},$$

where $X_{1,E}$ and $X_{2,C}$ denote the number of observed events in the experimental and the control group and $n_{1,E}$ and $n_{1,C}$ represent the first-stage sample sizes in the two groups. Blinded estimates of the event rates in each group can then be obtained by $\hat{p}_{C,A} = \hat{p}_0 - \Delta \cdot r / (1+r)$ and $\hat{p}_{E,A} = \hat{p}_0 + \Delta / (1+r)$. These estimates are used to recalculate the sample size. The benefit of the blinded recalculation is that the desired power can be maintained, even if the initially assumed value for p_0 was wrong.

It is well known that the chi-squared test in a fixed design does not maintain the nominal significance level, hence the same can be expected for a chi-squared test with a blinded sample size recalculation. In fact, Friede and Kieser (2004) showed that the actual levels of the test with and without recalculating the sample size are very close.

In a non-inferiority trial, the null and alternative hypothesis are

$$H_0 : p_E - p_C \leq -\delta, H_1 : p_E - p_C > -\delta,$$

where $\delta > 0$ is the fixed non-inferiority margin. The most commonly used test for this problem was

proposed by Farrington and Manning (1990). An approximate sample size formula for this test is

$$n = \frac{1+r}{r} \cdot \frac{\left(z_{1-\alpha/2}\sqrt{r \cdot \tilde{p}_C \cdot (1-\tilde{p}_C) + \tilde{p}_E \cdot (1-\tilde{p}_E)} + z_{1-\beta}\sqrt{r \cdot p_{C,A} \cdot (1-p_{C,A}) + p_{E,A} \cdot (1-p_{E,A})}\right)^2}{(\Delta+\delta)^2},$$

where $\tilde{p}_E$ and $\tilde{p}_C$ are large sample approximations of the restricted maximum likelihood estimators under the null hypothesis restriction $p_E - p_c = -\delta$ (see the Appendix of Farrington and Manning (1990) for the computation). The same formulas as for the chi-squared test can be used to estimate $\hat{p}_E$ and $\hat{p}_C$ in a blinded way, and these estimates have to be used to obtain blinded estimates $\hat{\tilde{p}}_E$ and $\hat{\tilde{p}}_C$ for the restricted maximum likelihood estimates. Plugging these estimates into the sample size formula gives the re-estimated sample size.

Like the chi-squared test, the Farrington-Manning test is also no exact test and can exceed the nominal significance level. Friede et al. (2007) showed that in general no further inflation of the type 1 error rate is caused by blinded re-estimation of the sample size. Nevertheless, it is possible for the chi-squared test as well as for the Farrington-Manning test to choose the nominal significance level smaller than α in order to protect the type 1 error rate at level α. Such an adjustment of α is implemented in **blindrecalc** for the binary and the continuous case.

3 Package structure

When a clinical trial with an internal pilot study is planned, it is essential to know the characteristics of the applied design. To this end, the performance in terms of achieved power levels, type I error rates, and sample size distribution has to be known for different values of the nuisance parameter and the first-stage sample size n_1. The package **blindrecalc** provides all necessary tools that are needed to plan a trial with a blinded sample size recalculation with only a small number of functions, which makes using the package very accessible.

blindrecalc makes use of R's S4 class system. This allows the application of the same methods for different design classes and facilitates the usage of the package. Furthermore, this approach makes the package easily extendable without any changes in the current source code.

The usage of **blindrecalc** is intended to be as intuitive as possible. To obtain characteristics of a blinded sample size recalculation procedure, two steps have to be made. At first, the user has to define a design object to indicate which test and which characteristics such as the desired type 1 and type 2 error rates are to be applied. To this end, the three functions `setupChiSquare`, `setupFarringtonManning`, and `setupStudent` exist to define a design object of the class corresponding to the respective test.

Secondly, the trial characteristic of interest can be calculated. Currently, the following methods are implemented: The method `toer` allows the computation of the actual type 1 error rate for different values of the nuisance parameter and the sample size of the internal pilot study. By means of `adjusted_alpha`, the adjusted significance level can be calculated that can be applied as nominal significance level when strict type 1 error rate control is desired. The method `pow` computes the achieved power of the design under a given set of nuisance parameters or internal pilot sample sizes. With `n_fix`, the sample size of the corresponding fixed design can be computed. Finally, the method `n_dist` provides plots and summaries of the distribution of the sample size. For all these methods (except for `n_dist`), the logical parameter `recalculation` allows to define whether a fixed design or a design with blinded sample size recalculation is analyzed.

4 Usage and example

For each test, there is a `setup` function (e.g., `setupChiSquare` for the chi-squared test) that creates an object of the class of the test. Each `setup` function takes the same arguments:

- `alpha`: The one-sided type 1 error rate.
- `beta`: The type 2 error rate.
- `r`: The allocation ratio between experimental and control group, with a default of 1.
- `delta`: The difference in effect size between alternative and null hypothesis.
- `alternative`: Whether the alternative hypothesis contains greater (default) or smaller values than the null.
- `n_max`: The maximal total sample size, with a default value of `Inf`.

In this example, the nuisance parameter is the overall response rate p. A difference in response rates between the two treatment groups of $\Delta = p_E - p_C = 0.2$ is to be detected. Using **blindrecalc**, a

chi-squared test that achieves a power of $1 - \beta = 0.8$ to detect this effect of $\Delta = 0.2$ and that uses a nominal type 1 error rate of $\alpha = 0.025$ can be set up by

```
design <- setupChiSquare(alpha = 0.025, beta = 0.2, delta = 0.2)
```

The sample size for a fixed design given one or multiple values of the nuisance parameter (argument `nuisance`) can then be calculated with the function `n_fix`:

```
n_fix(design, nuisance = c(0.2, 0.3, 0.4, 0.5))
#> [1] 124 164 186 194
```

The function `toer` calculates the actual level of a design with blinded sample size recalculation or of a fixed design (logical argument `recalculation`) given either one or more values of the total sample size in a fixed or the sample size for the first stage in a recalculation design (argument `n1`) or one or more values of the nuisance parameter. Note that all functions are only vectorized in one of the two arguments `n1` and `nuisance`. In this example, it is assumed that the internal pilot study contains half of the fixed sample size that would be needed if the overall response rate p equals 0.2. In this setting, **blindrecalc** can be used to compare the actual levels of a fixed design and a recalculation design with the same parameters.

```
n <- n_fix(design, nuisance = 0.2)
p <- seq(0.1, 0.9, by = 0.01)
toer_fix <- toer(design, n1 = n, nuisance = p, recalculation = FALSE)
toer_ips <- toer(design, n1 = n/2, nuisance = p, recalculation = TRUE)
```

In Figure 1, the type 1 error rate in dependence of the nuisance parameter is depicted for the designs with and without sample size recalculation. Note that, as mentioned in Section Binary outcomes, the level of significance exceeds the pre-defined level of $\alpha = 0.025$ in both cases. If strict control of the type 1 error rate is desired, the function `adjusted_alpha` can be used to to calculate an adjusted significance level, such that the nominal significance level is preserved.

```
adj_sig <- adjusted_alpha(design, n1 = n/2, nuisance = p, precision = 0.0001,
                          recalculation = TRUE)
design@alpha <- adj_sig
toer_adj <- toer(design, n1 = n/2, nuisance = p, recalculation = TRUE)
```

In this example, the adjusted significance level equals 0.0232 for the trial with internal pilot study, i.e., using this value as nominal level ensures that the actual significance level does not exceed $\alpha = 0.025$. Figure 1 demonstrates that the type 1 error rate is protected at level $\alpha = 0.025$ if the adjusted significance level is applied.

In the setting of binary outcomes, adjusting the level such that the nominal type 1 error rate is protected for any realization of the nuisance parameter in its domain $[0, 1]$ is feasible. However, when the nuisance parameter has an infinite domain, such as the variance in the case of continuous outcomes, this is not possible. The solution in these cases is to compute a $(1 - \gamma)$ confidence interval of the nuisance parameter in the blinded interim analysis and adjust the significance level such that the actual level is below $\alpha - \gamma$ for all values in this confidence interval (Friede and Kieser, 2011). If this approach is applied, the user can set the parameter `gamma`.

To calculate the power of either the internal pilot study design or the fixed design, the function `pow` can be used. Again, the function is vectorized in either `n1` or `nuisance`. This function can be used to compare the power values of the two designs under different actual values of the nuisance parameter.

```
pow_fix <- pow(design, n1 = n, nuisance = p, recalculation = FALSE)
pow_ips <- pow(design, n1 = n/2, nuisance = p, recalculation = TRUE)
```

As we can see in Figure 2, the power achieved by the internal pilot study design is very close to the target power of 0.8 in most cases. Only when the overall response rate is very close to 0 or 1, the power is exceeded. On the other hand, the fixed design is much more sensitive to the actual value of the nuisance parameter and the actual power can either be way too large or way too small if the sample size was calculated under wrong assumptions.

Finally, the distribution of the total sample size can be computed under different assumptions on the nuisance parameter with the function `n_dist`. This is particularly useful for the planning of internal pilot study designs since it allows the investigation of what could happen in a certain clinical trial and helps the applicant to prepare for different scenarios.

```
p <- seq(0.2, 0.8, by = 0.1)
n_dist(design, n1 = n/2, nuisance = p, plot = TRUE)
```

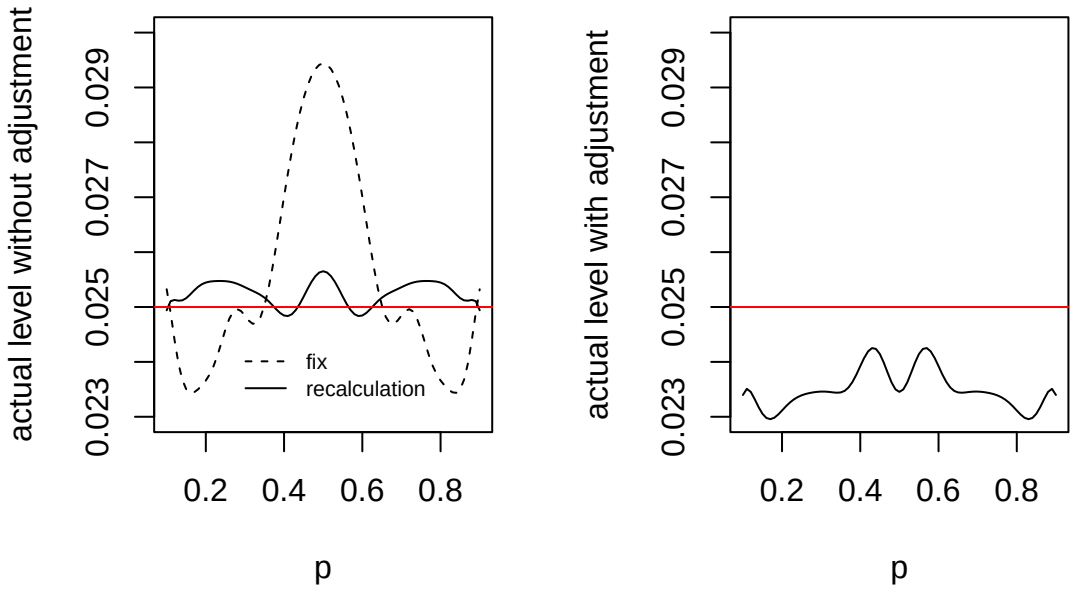


Figure 1: Actual significance level for different nuisance parameters with (right panel) and without (left panel) adjustment of the nominal significance level. In the adjusted case, the actual level is below the desired level (red line) for all nuisance parameters in contrast to the un-adjusted case.

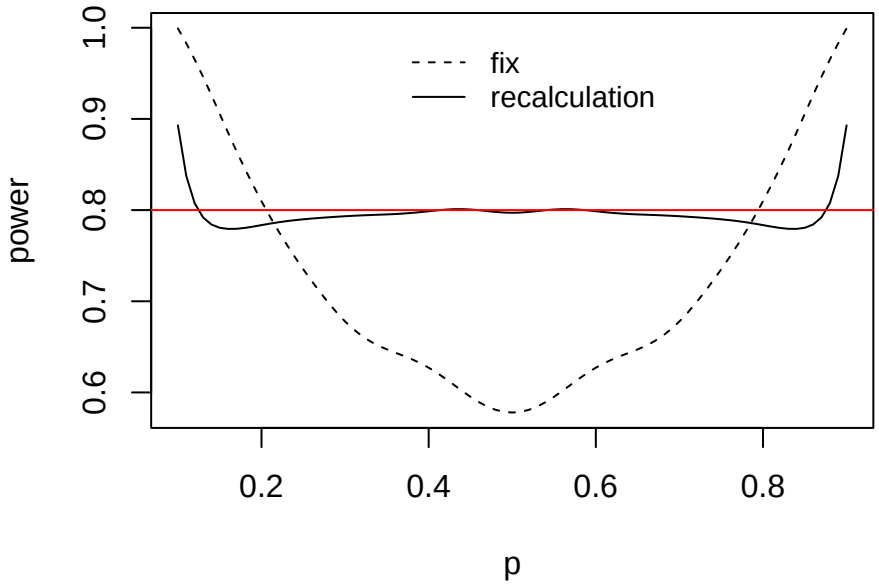


Figure 2: Power values for different nuisance parameters for a fixed design and a design with blinded sample size recalculation. The design with recalculation meets the target power (red line) for a wider range of nuisance parameters.

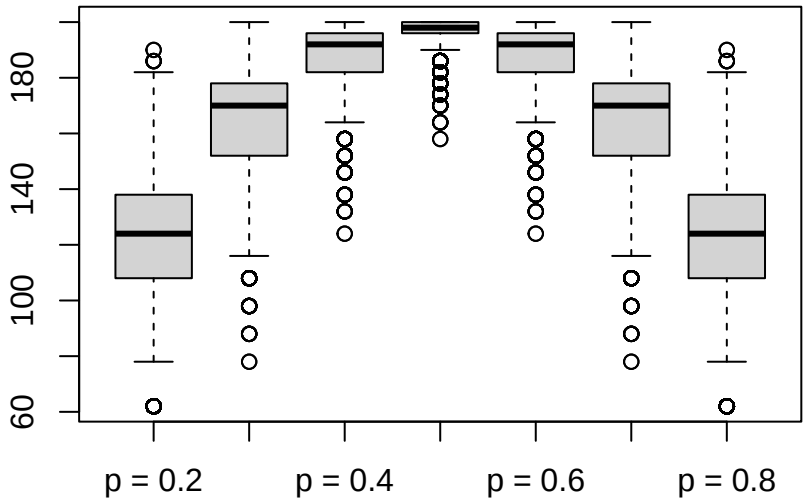


Figure 3: Distribution of the sample size for a design with blinded sample size recalculation in dependence of the nuisance parameter. For more extreme values of the nuisance parameter the variance of the sample size distribution becomes larger.

```
#>         p = 0.2  p = 0.3  p = 0.4  p = 0.5  p = 0.6  p = 0.7 p = 0.8
#> Min.     62.000  78.0000 124.0000 158.0000 124.0000  78.0000  62.000
#> 1st Qu. 108.000 152.0000 182.0000 196.0000 182.0000 152.0000 108.000
#> Median  124.000 170.0000 192.0000 198.0000 192.0000 170.0000 124.000
#> Mean    125.021 164.6057 188.4801 196.5075 188.4801 164.6057 125.021
#> 3rd Qu. 138.000 178.0000 196.0000 200.0000 196.0000 178.0000 138.000
#> Max.    190.000 200.0000 200.0000 200.0000 200.0000 200.0000 190.000
```

By default, `n_dist` prints a summary of the sample size distribution for each nuisance parameter. With `plot = TRUE`, a series of boxplots is drawn (cf. Figure 3). Since the maximum sample size is obtained if the overall response rate is estimated to be 0.5 in the sample size recalculation, this maximum can occur under any true value of the nuisance parameter (except for 0 and 1), albeit with very small probability. For this reason, sample sizes that occur with a probability of less than 0.01% are ignored. This is not the case for continuous outcomes since there, the sample size distributions are determined by simulation.

For continuous outcomes, i.e., the (shifted) *t*-test, the functional content of **blindrecalc** is the same as in the binary case that was presented in this example. The only difference is that in the continuous case, the numbers are computed by simulation. Thus, the user can set the parameters `iters`, defining the number of simulation iterations, and `seed`, the random seed for the simulation.

5 Development principles

The utilization of R's object-oriented programming capabilities implies that the example that was presented for the chi-squared test could very similarly be applied to the Farrington-Manning test or the *t*-test. Besides using S4 classes, the following development principles of **blindrecalc** should be briefly described.

All calculations for binary outcomes are exact and require nested for-loops. Since for-loops are known to be very slow in R, all computation-intensive functions for the chi-squared test and the Farrington-Manning test are implemented in C++ via the **Rcpp** package (Eddelbuettel and François, 2011) to speed up the calculations significantly.

blindrecalc is developed open-source on GitHub.com. The entire source code can be found at `https://github.com/imbi-heidelberg/blindrecalc`. This allows anyone to contribute to **blindrecalc** and, furthermore, provides maximal transparency. To ensure a certain quality of the provided code, **blindrecalc** is checked by unit tests using the package **covr** (Hester, 2020). The unit tests compare numbers for the sample size, type 1 error rate, and power calculated with **blindrecalc** with numbers from peer-reviewed publications and, furthermore, check the technical functionality of the package such as vectorization and display of error messages. Thus, the unit tests do not only monitor the technical accuracy of the package's results but also their content-related correctness. The current version **blindrecalc** 0.1.3 achieves a code coverage of 100%, i.e., each line of the source code is checked by at least one unit test.

6 Conclusion

In this paper, we introduced the R-package **blindrecalc** that can be used to plan clinical trials with a blinded sample size recalculation in an internal pilot study design when either continuous or binary outcomes in a superiority or non-inferiority setting are of interest. We introduced the basic methodology of internal pilot studies and explained how the package can be used to calculate the operating characteristics of a trial with such a design.

The scope of **blindrecalc** can simply be extended due to its modular character. Blinded sample size recalculation can be applied to many different types of clinical trials. For instance, there exists research on further kinds of outcomes (e.g., see Friede and Schmidli (2010) for count data) or on different study designs (e.g., see Golkowski et al. (2014) for bioequivalence trials). The implementation of internal pilot studies for such cases in **blindrecalc** is an exciting area of future work.

Bibliography

M. A. Birkett and S. J. Day. Internal pilot studies for estimating sample size. *Statistics in Medicine*, 13 (23-24):2455–2463, 1994. URL https://doi.org/10.1002/sim.4780132309. [p137, 138]

Committee for Medical Products for Human Use (CHMP). Reflection paper on methodological issues in confirmatory clinical trials planned with an adaptive design. https://www.ema.europa.eu/en/documents/scientific-guideline/reflection-paper-methodological-issues-confirmatory-clinical-trials-planned-adaptive-design_en.pdf, 2006. Accessed: March 10, 2022. [p137]

J. S. Denne and C. Jennison. Estimating the sample size for a t-test using an internal pilot. *Statistics in Medicine*, 18(13):1575–1585, 1999. URL https://doi.org/10.1002/(SICI)1097-0258(19990715)18:13<1575::AID-SIM153>3.0.CO;2-Z. [p137]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL https://doi.org/10.18637/jss.v040.i08. [p143]

C. P. Farrington and G. Manning. Test statistics and sample size formulae for comparative binomial trials with null hypothesis of non-zero risk difference or non-unity relative risk. *Statistics in Medicine*, 9(12):1447–1454, 1990. URL https://doi.org/10.1002/sim.4780091208. [p140]

T. Friede and M. Kieser. A comparison of methods for adaptive sample size adjustment. *Statistics in Medicine*, 20(24):3861–3873, 2001. URL https://doi.org/10.1002/sim.972. [p139]

T. Friede and M. Kieser. Blinded sample size reassessment in non-inferiority and equivalence trials. *Statistics in Medicine*, 22(6):995–1007, 2003. URL https://doi.org/10.1002/sim.1456. [p139]

T. Friede and M. Kieser. Sample size recalculation for binary data in internal pilot study designs. *Pharmaceutical Statistics*, 3(4):269–279, 2004. URL https://doi.org/10.1002/pst.140. [p137, 139]

T. Friede and M. Kieser. Sample size reassessment in non-inferiority trials. *Methods of Information in Medicine*, 50(3):237–243, 2011. URL https://doi.org/10.3414/ME09-01-0063. [p141]

T. Friede and H. Schmidli. Blinded sample size reestimation with count data: methods and applications in multiple sclerosis. *Statistics in Medicine*, 29(10):1145–1156, 2010. URL https://doi.org/10.1002/sim.3861. [p144]

T. Friede, C. Mitchell, and G. Müller-Velten. Blinded sample size reestimation in non-inferiority trials with binary endpoints. *Biometrical Journal*, 49(6):903–916, 2007. URL https://doi.org/10.1002/bimj.200610373. [p137, 140]

D. Golkowski, T. Friede, and M. Kieser. Blinded sample size re-estimation in crossover bioequivalence trials. *Pharmaceutical Statistics*, 13(3):157–162, 2014. URL https://doi.org/10.1002/pst.1617. [p144]

A. L. Gould. Interim analyses for monitoring clinical trials that do not materially affect the type I error rate. *Statistics in Medicine*, 11(1):55–66, 1992. URL https://doi.org/10.1002/sim.4780110107. [p137]

J. Hester. *covr: Test Coverage for Packages*, 2020. URL https://CRAN.R-project.org/package=covr. R package version 3.5.1. [p143]

M. Kieser. *Methods and Applications of Sample Size Calculation and Recalculation in Clinical Trials*. Springer Series in Pharmaceutical Statistics. Springer Verlag, 2020. URL https://doi.org/10.1007/978-3-030-49528-2. [p139]

M. Kieser and T. Friede. Re-calculating the sample size in internal pilot study designs with control of the type I error rate. *Statistics in Medicine*, 19(7):901–911, 2000. URL https://doi.org/10.1002/(SICI)1097-0258(20000415)19:7<901::AID-SIM405>3.0.CO;2-L. [p137]

M. Kieser and T. Friede. Simple procedures for blinded sample size adjustment that do not affect the type I error rate. *Statistics in Medicine*, 22(23):3571–3581, 2003. URL https://doi.org/10.1002/sim.1585. [p139]

K. Lu. Distribution of the two-sample t-test statistic following blinded sample size re-estimation. *Pharmaceutical Statistics*, 15(3):208–215, 2016. URL https://doi.org/10.1002/pst.1737. [p139]

K. Nakata, S. Shikata, T. Ohtsuka, T. Ukai, Y. Miyasaka, Y. Mori, V. V. D. M. Velasquez, Y. Gotoh, D. Ban, Y. Nakamura, Y. Nagakawa, M. Tanabe, Y. Sahara, K. Takaori, G. Honda, T. Misawa, M. Kawai, H. Yamaue, T. Morikawa, T. Kuroki, Y. Mou, W.-J. Lee, S. V. Shrikhande, C. N. Tang, C. Conrad, H.-S. Han, P. Chinnusamy, H. J. Asbun, D. A. Kooby, G. Wakabayashi, T. Takada, M. Yamamoto, and M. Nakamura. Minimally invasive preservation versus splenectomy during distal pancreatectomy: a systematic review and meta-analysis. *Journal of Hepato-Biliary-Pancreatic Sciences*, 25(11):476–488, 2018. URL https://doi.org/10.1002/jhbp.569. [p137]

C. Stein. A two-sample test for a linear hypothesis whose power is independent of the variance. *Annals of Mathematical Statistics*, 16(3):243–258, 1945. URL https://doi.org/10.1214/aoms/1177731088. [p137]

J. Wittes and E. Brittain. The role of internal pilot studies in increasing the efficiency of clinical trials. *Statistics in Medicine*, 9(1-2):65–72, 1990. URL https://doi.org/10.1002/sim.4780090113. [p137, 138]

D. M. Zucker, J. T. Wittes, O. Schabenberger, and E. Brittain. Internal pilot studies II: comparison of various procedures. *Statistics in Medicine*, 18(24):3493–3509, 1999. URL https://doi.org/10.1002/(SICI)1097-0258(19991230)18:24<3493::AID-SIM302>3.0.CO;2-2. [p139]

7 Contribution

The first two authors contributed equally to this manuscript.

Lukas Baumann
Institute of Medical Biometry
University of Heidelberg
Im Neuenheimer Feld 130.3
69120 Heidelberg
Germany
ORCiD 0000-0001-7931-7470
baumann@imbi.uni-heidelberg.de

Maximilian Pilz
Institute of Medical Biometry
University of Heidelberg
Im Neuenheimer Feld 130.3
69120 Heidelberg
Germany
ORCiD 0000-0002-9685-1613
pilz@imbi.uni-heidelberg.de

Meinhard Kieser
Institute of Medical Biometry
University of Heidelberg
Im Neuenheimer Feld 130.3
69120 Heidelberg
Germany
ORCiD 0000-0003-2402-4333
kieser@imbi.uni-heidelberg.de

Revisiting Historical Bar Graphics on Epidemics in the Era of R ggplot2

by Sami Aldag, Dogukan Topcuoglu and Gul Inan

Abstract This study is motivated by an article published in a local history magazine on "Pandemics in the History". That article was also motivated by a government report involving several statistical graphics which were drawn by hand in 1938 and used to summarize official statistics on epidemics occurred between the years 1923 and 1937. Due to the aesthetic information design available on these historical graphs, in this study, we would like to investigate how graphical elements of the graphs such as titles, axis lines, axis tick marks, tick mark labels, colors, and data values are presented on these graphics and how to reproduce these historical graphics via well-known data visualization package **ggplot2** in our era.

1 Introduction

In August 2018, a local history journal named "Social History" published an issue on "Pandemics in the History" which left a deep effect on the world, created new public policies, and, in turn, reshaped state-society relations over the globe (Toplumsal Tarih, 2018). The issue involves several articles specifically on the pandemics such as plague, malaria, cholera, diphtheria, trachoma, syphilis, and tuberculosis, where the content of the articles were accompanied by rich historical photographs and visualizations.

The article entitled "Fight against syphilis that forgot to embrace in the era of early Republic" by Malkoc (2018) in this issue specifically took our attention since this article involves several aesthetically attractive statistical column bar graphics, which were assumed to be drawn by hand with the help of a ruler, with a citation to a government report published in 1938. A deep investigation of this 620-page government report, which is also available online at `https://acikerisim.tbmm.gov.tr/handle/11543/553`, revealed that it has a section where the Ministry of Health reported official statistics related to all health policy actions taken and health services provided to improve the public health between the years 1923 and 1937. While most of the official statistics were summarized in tabular form, around forty different statistical bar graphics were also used to visually summarize the official statistics related to various epidemic diseases such as smallpox, trachoma, malaria, and syphilis which occurred in the country between the years 1923 and 1937. While doing so, it was obvious that the government officials put a special emphasis on the information design on the graphics at that time. A further investigation through discussions with several academics studying on the history of graphic design also revealed that using aesthetically designed statistical graphics were already common in the country in late 1800's parallel to the globe (Durmaz, 2017). Here we note that well-known early examples of visualization of official statistics over the globe include Statistical Atlas of the United States in late 1800's, Album de Statistique Graphique, and Graphical Statistical Atlas of Switzerland 1897–2017 where selected illustrative examples are available in Friendly (2008).

Furthermore, statistical graphics were also used by the Goverment officials as an effective communication tool to inform the society who had low literacy skills during that period (Sengul, 2017). This argument is still true during the Covid-19 pandemic. With the help of technological advances in data visualization software in our era, government officials, authorities, and media intensively use (mostly interactive) data visualization tools to release pandemic related statistics to the public in a very short time to keep the society informed (e.g., please visit GitHub account of the Civil Protection Department of the Italian government given at del Consiglio dei Ministri (2020) and the GIS based interactive dahsboard of Coronavirus Resource Center at Johns Hopkins University (2020)). Hence, as in the past, during the Covid-19 pandemic over the globe, data visualization continues to be the most effective way of sharing information and informing society (McCoy, 2020).

On the other hand, while statisticians and graphic designers may have different priorities on what makes a good graphic (Gelman and Unwin, 2013; Quito and Kopf, 2020), reading graphics, understanding the information design behind them and interpreting them require practice of data literacy for the society (e.g., use of semi logarithmic graphs for visualizing rate of change of Covid-19 infections has been a long discussion (Garthwaite, 2020)). In this sense, motivated by i) VanderPlas et al. (2019) who revisited, reinterpreted, and reproduced some novel charts from 1870 Statistical Atlas with moden technology, ii) the exhibition entitled "Speak to the Eyes", curated by Durmaz (2017) which revisited and turned some historical graphics on justice statistics in 1920's into motion graphics, and iii) Matthew (2019) who revisited and reproduced W.E.B. Du Bois'in visualizations on social and economic life of African-Americans in 1900's via `R`, in this study, we would like to revisit and

reproduce the historical column bar graphics used to visualize official statistics on epidemics occurred in our country between the years 1923 and 1937 via R. For that reason, the aim of this study is to investigate i) how graphical elements of the historical column bar graphs such as titles, axis lines, axis tick marks, tick mark labels, bar colors, and data values are presented on these graphics and ii) how to reproduce these 1938-made and hand-drawn graphics via well-known data visualization software **ggplot2** (Wickham et al., 2020) in our era.

The subsequent sections of the paper are organized as follows: We give general information on the graphical elements of column bar graphics and we talk about the column bar graphics used in this study. Then we also give redesigned versions some of the selected historical graphics. Finally, we finish with some concluding remarks.

2 An overview on graphical elements of a column bar

The bar chart was first invented by William Playfair to visualize the imports and exports of Scotland between seventeen countries in year 1871 and was first published in his book entitled "Commercial and Political Atlas" in 1876 (please visit Figure C in Beniger and Robyn (1978)). In a general sense, column bar graphics are a statistical visualization technique used to present quantitative information through a series of vertical rectangles. They are mostly used to display and compare data values of multiple groups over time (Harris, 2000). Column bars mostly have a quantitative linear scale on the *vertical axis*. The height of each column in a bar graph is proportional to the numerical value it represents so that the viewer make a visual comparison between the columns. When the vertical axis is not available in the graph, the *actual data value* which each column represents can be either placed inside the column or at the top of the column. *Alignment* of the data value can be done horizontally or vertically, depending on the space available on the graph.

The scale on the *horizontal axis* is generally categorical or sequential (e.g. time series) and *tick marks* may or may not be used on the horizontal axis. The *width* of columns and the *spacing* between the columns are generally kept uniform over columns in a graph. The data series belonging to different groups are generally differentiated with each other by assigning different *colors* or *patterns* to the groups. The differentiation in *colors* and/or *patterns* are also reflected into the *legend keys* to help the viewer to identify the quantitative information displayed in the graph. Furthermore, the information on the legend keys is ordered as it appears on the graph. The *legends* can be placed anywhere on the graph, but the closer to the information they represent, the more convenient for the viewer to decode the information on the graph. *Grid lines* at the background are not generally preferred since rectangular bars are very dominant visual objects. The *background color* may contrast the color of the columns to increase communication between the graph and the viewer. We illustrate these graphical elements in Figure 1.

3 Column bar graphics used in this study

Due to World War I (1914-1918) and then Independence War (1919-1922), the country, which was founded in 1923, had to simultaneously deal with many infectious diseases such as smallpox, malaria, plague, syphilis, trachoma, tuberculosis, leprosy, and typhus. Due to the increasing number of infectious diseases and infected people, the government had to develop new public health policies and offer health care services through launching new hospitals, training health care workers (including medical doctors, nurses and so on), and producing disease diagnostic kits, drugs, serum, and vaccines. In spite of many impossibilities, the government had achieved great success in prevention of infectious diseases during the period of 1923-1937. In 1938, all the efforts, especially the ones on the workload of hospitals and then on vaccine administration in the country, were summarized officially and these official statistics were visualized through statistical column bar graphics along with the tabular raw data in the government report. We should note that the government report does not provide any additional information or explanation related to these graphics.

In this study, among these historical column bar graphics, we investigated and reproduced nine of them. We provide the original graphics alongside the reproduced graphics as well. In this sense, we categorize them into five main parts with respect to the number of data series available as well as grouping structure of the bars (e.g., overlapped, side-by-side, and paired bar graphs). We also kindly invite readers to look at the R codes available as a Supplementary material while investigating the graphics.

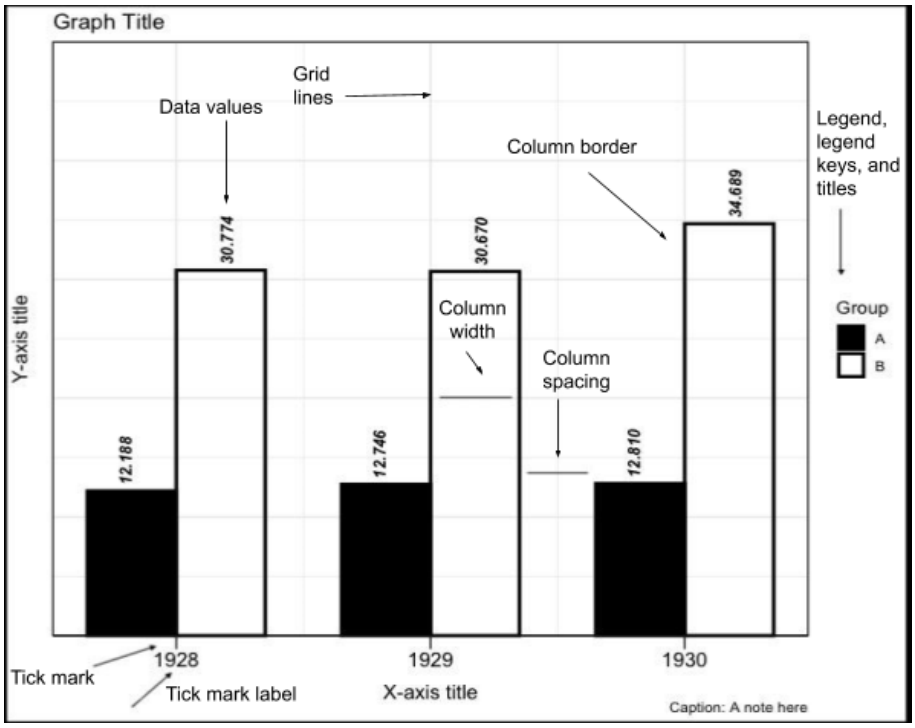


Figure 1: An anatomy of a column bar graph.

Bar graph with one data series

In the bar graphs with one data series, bars are used to compare a single numerical variable per item or category. Figure 2 gives the amount of smallpox vaccine administered in various regions of the country between the period 1925 and 1937. Smallpox is a deadly infectious disease accompanied by lesions filled with thick liquid appearing on the face, mouth, nose, and body of a person. Within the early days of exposure, it had been shown that the vaccination can prevent or lessen the severity of the disease. For that reason, the vaccination was mandatory for new borns, at schools, and at some workplaces. Note that these vaccines were distributed for free to prevent the disease.

In Figure 2, we can see that the background color of the figure is white. There is no vertical axis and related information on the vertical axis (e.g., axis line, axis title, axis tick marks, and tick mark labels). We can get the frequencies of each column bar through the data values placed inside the columns. Consequently, the column bar heights are directly proportional to the data values they represent. Since the height of the columns are taller and take space in the figure plotting area, the data values are placed vertically inside the columns. The horizontal axis refers to the time interval with linear increments without having an axis title. Due to the white background color of the figure, the column bars are filled in with black color whereas the data values are colored in white for contrast. Due to a large number of column bars and lack of space, bar widths and the spacing between columns are kept short and the labels of the horizontal axis tick marks are displayed vertically.

Since the heights of the columns of the graph are directly proportional to the data value they represent, the `geom_col()` layer right after the main `ggplot()` call in **ggplot2** is used to produce Figure 3. The data values are placed onto the graphic via an `annotate()` layer. The white background is obtained via `theme_classic()` layer. The structure of the graph is mostly obtained through modifying the components of `theme()` layer such as `axis.line`, `axis.title`, `axis.ticks`, and `axis.text` in **ggplot2**, in addition to `geom_col()` layer. The main figure title consists of four lines. However, the first three line and the last line of the title have different font types, sizes, and faces (i.e., italic and unitalic texts). For that reason, several `annotate()` layers are further used to run the full title rather than `ggtitle()` or `labs()` layer which assumes a uniform text structure over the multiple lines. Finally, we can see that the number of smallpox vaccines administered increased over the years.

Overlapped bar graphs with two data series

Figure 4 presents the service of hospitals and dispensaries within the Department of Control of Trachoma between the years 1925 and 1937 with respect to the number of inpatient treatments performed (in black) and the number of surgeries performed for treating trachoma (in white).

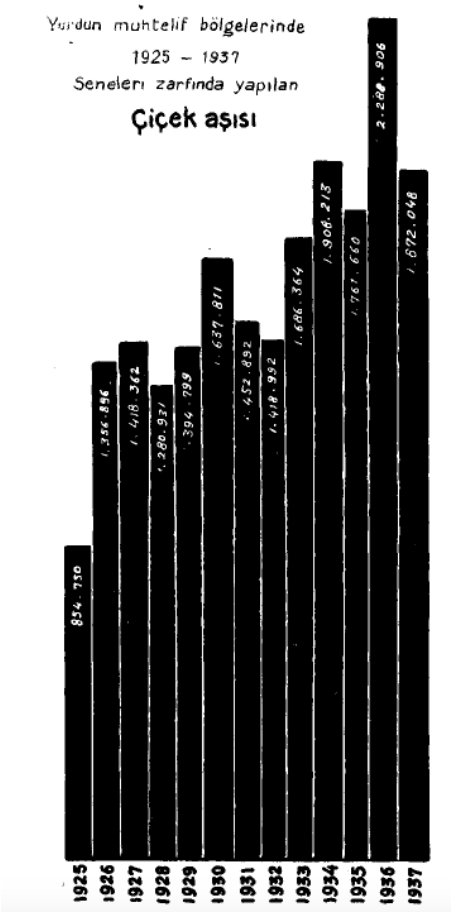


Figure 2: Historical figure on "Smallpox vaccine administered in various regions of the country, between the period 1925-1937" retrieved from https://acikerisim.tbmm.gov.tr/handle/11543/553.

Trachoma is an infectious eye disease caused by a bacteria and is transmitted among humans through shared use of items used for cleaning face. If it is not treated at the earlier stages, it may lead to damages in eye cornea or even to blindness. In the early stages of trachoma, antibiotics may be effective to eliminate the infection, whereas surgery may be required at the later stages.

As in Figure 2, the background color of the Figure 4 is white and there is no vertical axis and any information related to the vertical axis (e.g., axis line, axis title, axis tick marks, and tick mark labels). The columns of both groups are overlapped 100% and the column bar heights are directly proportional to the data values they represent. The data set for the number of inpatient treatments is always shorter than the data set for the number of surgeries performed for treating trachoma over the years 1925 and 1937. Thus, the columns for inpatient treatments are positioned in front of the columns for the number of surgeries performed. However, the disparity between the heights of both groups is manipulated through assigning a strong color, black, to the number of inpatient treatments, and a recessive color, white, to the number of outpatient treatments, which is a common strategy in graphic design (White, 1984). This emphasis is also reflected in the legend keys such that the legend keys are ordered according to color, not alphabetically.

The data values for the surgery group between the years 1925 and 1931 are placed vertically at the top of the columns. Those between the years 1932 and 1937 are placed inside the columns vertically due to lack of space in the plotting region, whereas the data values for the number of inpatient treatments are always placed vertically inside the column. All the data values for each group are in black since the background color is white. Due to the reasonable number of column bars in the plotting area, the width of the column bars and the inter-bar spacing between them are now increased.

Unlike Figure 2, there are no labels for the horizontal axis tick marks now. However, the third line of main graph title gives the clue that horizontal axis starts from 1925 and goes to 1937. As the reviewer pointed out, we think that horizontal tick mark labels here were unintentionally forgotten since this is the only bar graph with missing horizontal tick mark labels in the government report. However, this may result in a cognitive effort for the viewer if the viewer would like to know the exact information for the number of inpatient treatments and/or the number of surgeries performed

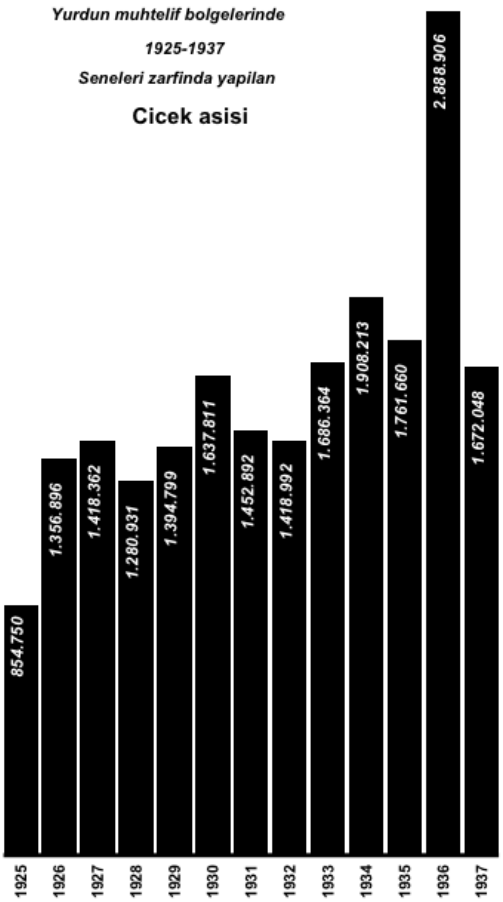


Figure 3: Reproduced figure on "Smallpox vaccine administered in various regions of the country, between the period 1925-1937".

throughout the years, especially for the years in the middle of 1925 and 1937. In that case, the viewer has to count the number of bars to get precise information. On the other hand, if the interest is on the overall trend of the number of inpatient treatments and/or the number of surgeries performed over the years, then comparing the heights of the bars visually will eliminate this problem. If there is not enough space to place all the tick marks on the horizontal axis, then two design choices can be followed here: 1) placing all the labels with some vertical shift such as 90 degree or 45 degree alignment, or 2) starting labeling at the year 1925 and labeling the years with one year apart.

In Figure 5, the 100% overlapping structure of column bars is obtained via setting argument `position = "identity"` in `geom_col()` layer. Note that the look of Figure 5 requires arrangement of the levels of grouping factor in the data with order of inpatient treatments and surgeries performed, respectively. This grouping variable is also mapped into `fill` and `alpha` arguments of the aesthetics of the main `ggplot()` call since the fill-in colors of the bars and transparency level of the bars should be matched with the levels of this grouping variable. In addition to modifying several components of `theme()` layer, assigning "black" color to the inpatient treatments and "white" color to the surgeries performed via `scale_fill_manual()` layer, and then assigning low level of transparency "1" to the black color of inpatient treatments and high level of transparency "0" to the white color of surgeries performed via `scale_alpha_manual()` layer would yield the final look of the figure. Hence, the order of elements of the vector of colors in `scale_fill_manual()` layer and the order of elements of the vector of transparency in `scale_alpha_manual()` layer are matched with the order of the levels of the grouping variable. Lastly, if transparency were not added to the plot, the color of the second level of the grouping variable will be displayed only due to the overlapping structure of the column bars.

The white line segment in the first column of Figure 5 is integrated via an `annotate()` layer along with `rect` argument. On the other hand, three-lined main graph title is run with `labs()` and `annotate()` layers due the italicized font structure of the middle line compared to the unitalicized font structure of the first and the last lines. Lastly, we can say that both the number of inpatient treatments and the number of surgeries performed increased considerably over the years.

Figure 6 shows the service of the Zonguldak Government Hospital between the years 1924 and 1937

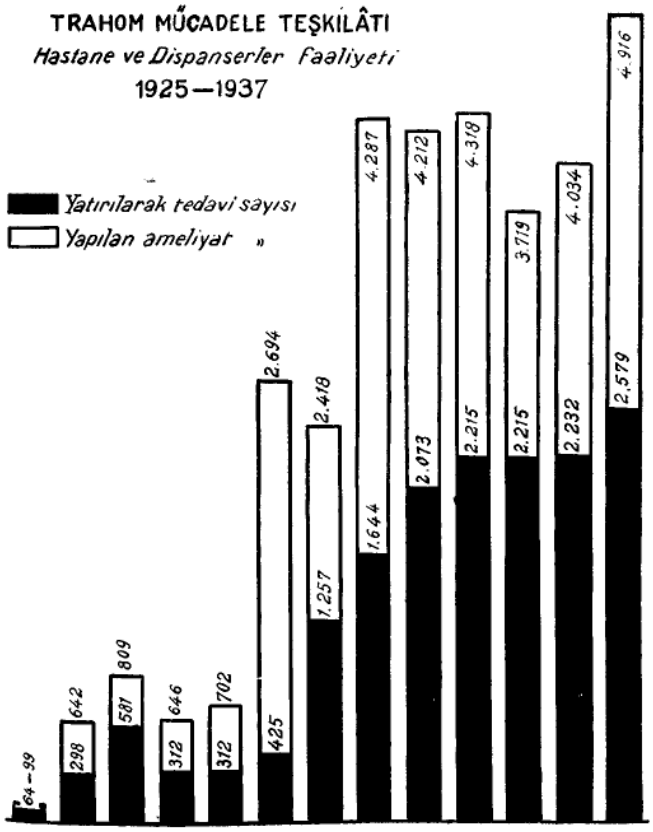


Figure 4: Historical figure on "The service of hospitals and dispensaries within the Department of Control of Trachoma, 1925-1937" retrieved from https://acikerisim.tbmm.gov.tr/handle/11543/553 (■ The number of inpatient treatments □ The number of surgeries performed).

with respect to the number of inpatient treatments (in black) and the number of outpatient treatments (in white). The data value for the number of outpatient treatment is not available in 1924 and is coded as **NA** in the data. While the columns for the number of inpatient treatments are taller than that of outpatient treatment over the period 1925 and 1932, the columns for the number of outpatient treatments are taller than that of inpatient treatments over the period 1933 and 1937. To increase the dominance of the number of inpatient treatments over the number of outpatient treatments, the former group is colored in black.

In year 1932, the number of outpatient treatments is close to the number of inpatient treatments in magnitude, where the frequencies are 725 and 815 respectively. Since there is not enough space to place the number 725 vertically inside the bars, it is aligned horizontally and colored in white due to the black background color. To be consistent with white front-positioned outpatient treatment bars, all the data labels of outpatient treatments are placed horizontally between 1925 to 1932 and then vertically onwards. This approach gives a practical information design approach.

The R code structure of Figure 7 is very challenging and is not straightforward since `geom_col(position = "identity")` assumes that difference between two data series over the years takes a uniform behaviour, i.e., it assumes that a series is always either greater than the other one, or always less than the other one. If it is not so, `geom_col(position = "identity")`, `scale_fill_manual()`, and `scale_alpha_manual()` layers cannot handle the zig-zag pattern in the difference of data series when drawing bars. This problem actually opens a research door for **ggplot2**.

Figure 8 represents the workload of private hospitals between the years 1926 and 1937 with respect to the number of inpatient treatments (in black) and the number of outpatient treatments (in white). Except the year 1926, the columns for the number of inpatient treatments are always shorter than the columns for the number of outpatient treatments. In 1926, the number of inpatient treatments is 17,700, which is greater then the number of outpatient treatments, 16,009. The columns, which are positioned in the front, are shifted to the left, resulting in around 70% overlapping. Due their front position and stronger color, the number of inpatient treatments take the attention of the viewer. On the other hand, since the data values for the number of outpatient treatments is just twice of the data values for the number of inpatient treatments in the years 1927 and 1928, the heights of the column bars are close to each other and hence there is not enough space to place the data values for the number of inpatient

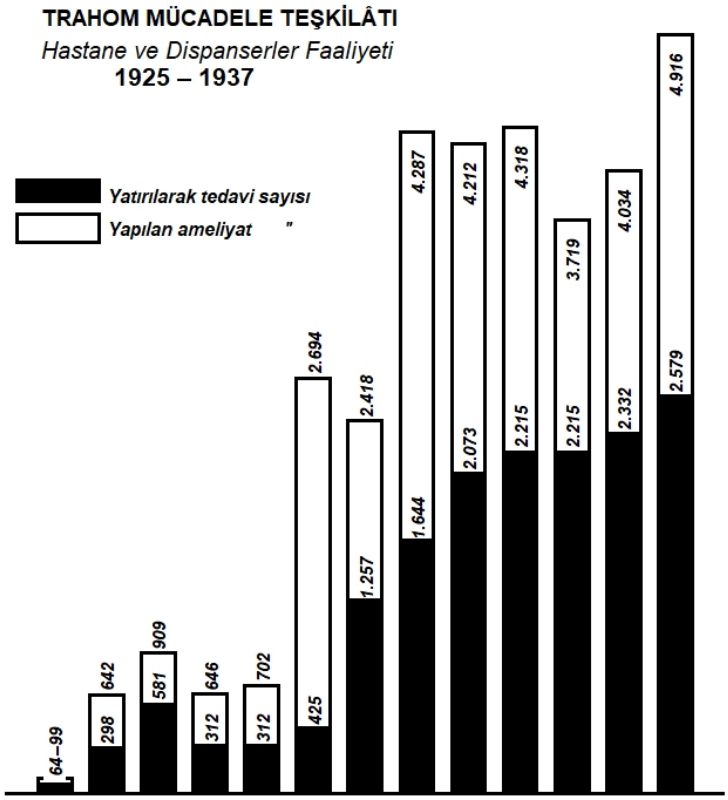


Figure 5: Reproduced figure on "The service of hospitals and dispensaries within the Department of Control of Trachoma, 1925-1937" (■ The number of inpatient treatments □ The number of surgeries performed).

treatments in the column bars vertically. To avoid overlapping of two data labels, the front data labels are placed horizontally, which is a practical approach. This information gives us an another idea that alignment of data values according to space available in Figures 6 and 8 is reasonable. However, as the reviewer pointed out, this inconsistencies i.e., aligning texts horizontally then vertically, is visually distracting and may decrease the readability.

The `R` code structures of Figures 5 and 9 are very similar to each other. The 70% overlapping position of columns bars in Figure 9 is adjusted through setting `position = position_dodge(width = 0.3)` in `geom_col()` layer. Furthermore, in year 1926, the column of outpatient treatment is less than that of inpatient treatment, which is vice versa in the subsequent bars. As we discussed a similar problem for Figure 7, `geom_col(position = "identity")` could not handle this and the white rectangular column bar in year 1926 is drawn manually via `annotate()` layer with `rect` argument.

In Figures 4, 6, and 8, the legend is placed vertically at the top-left of the plotting area, since the overall structure of the graphs are left-skewed and the legend keys are ordered according to color, not alphabetically. Furthermore, in Figures 4-9, the double apostrophe, " , in the title of second legend key is a typographic symbol, called a ditto mark, which is used for repeated words above it. In hand-written texts, ditto mark is used to save time and effort from the writer. Since the government report includes around forty figures, this may be the reason why ditto mark is also used in the legend key titles. While this approach also reduces the amount of ink used in the figures, it does not distort the readability of the graph. However, we should note that in today's technology, the repetitive words can be typed as needed with less effort.

We would like to note that the Figures 4, 6, and 8 may look like as if they were stacked bar graphs. However, since the government report published in 1938 includes the raw tabular data in it, we are 100% confident to say that these figures are overlapped bar graphs. Furthermore, we would also like to note that in stacked bar charts, the height of the column bar gives the total frequency of all groups on a given year. If one needs the frequency of a specific group on that year, then an additional arithmetic operation such as subtraction should be done to calculate it. Keeping in mind that these graphs are

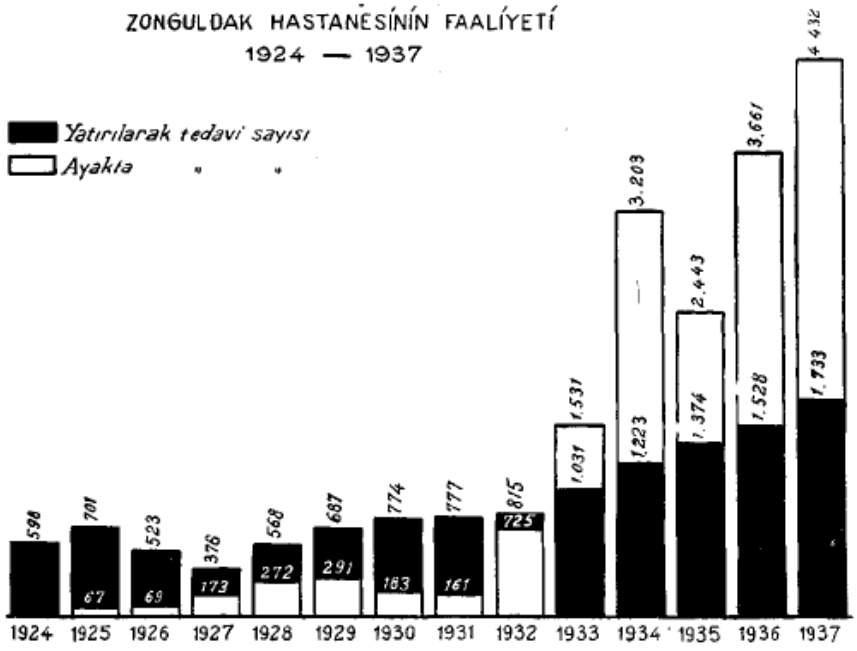


Figure 6: Historical figure on "The service of the Zonguldak Government Hospital, 1924-1937" retrieved from https://acikerisim.tbmm.gov.tr/handle/11543/553 (■ The number of inpatient treatments □ The number of outpatient treatments).

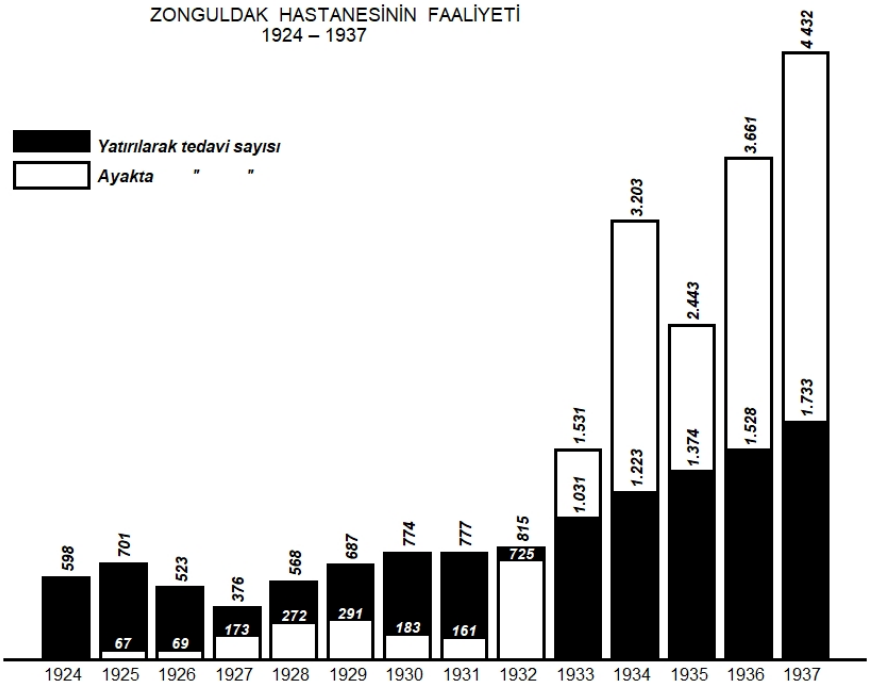


Figure 7: Reproduced figure on "The service of the Zonguldak Government Hospital, 1924-1937" (■ The number of inpatient treatments □ The number of outpatient treatments).

presented to the public who has a low level of literacy, doing an extra mathematical work in stacked bar charts would add a computational burden to the viewer, which may not be possible for the viewer at that time.

Another distinction between overlapped bar graphs and stacked bar graphs is that overlapped bar graphs are used to display the comparison between two closely related numerical variables over an

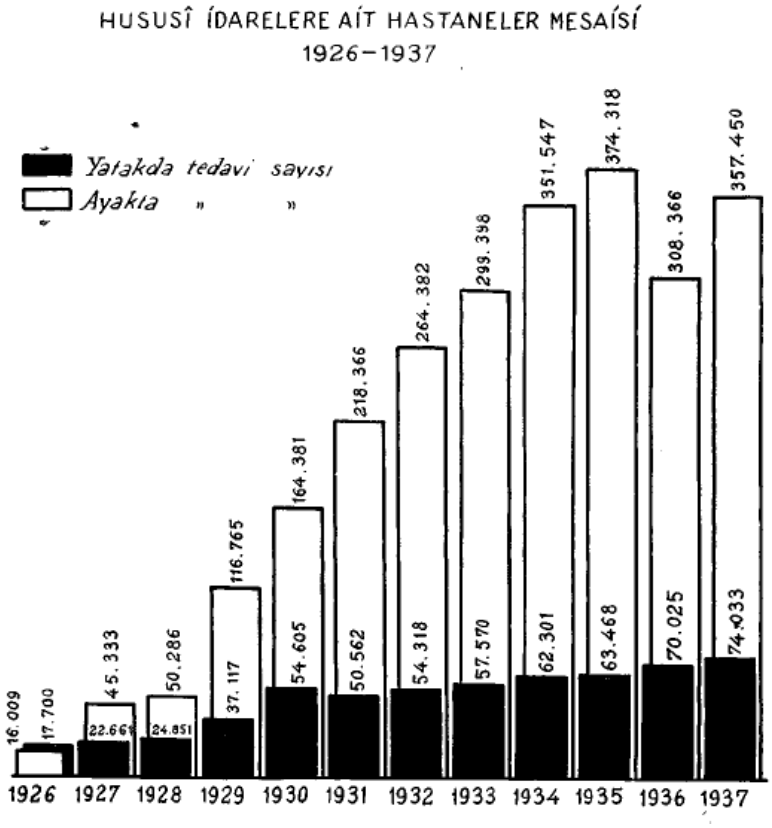


Figure 8: Historical figure on "The workload of private hospitals, 1926-1937" retrieved from https://acikerisim.tbmm.gov.tr/handle/11543/553 (■ The number of inpatient treatments □ The number of outpatient treatments).

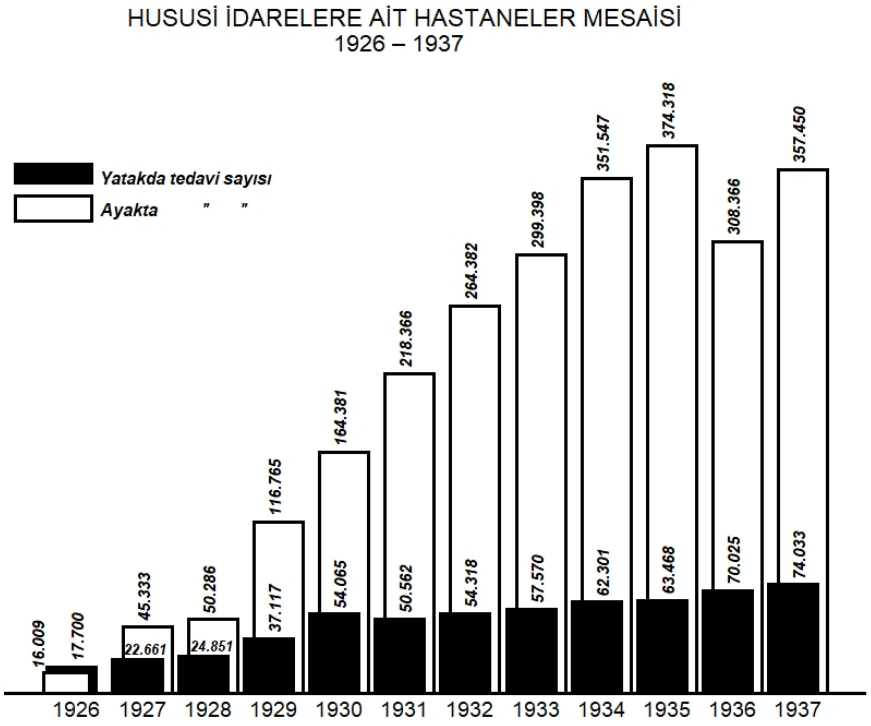


Figure 9: Reproduced figure on "The workload of private hospitals, 1926-1937" (■ The number of inpatient treatments □ The number of outpatient treatments).

item/category (here it is years), whereas stacked bar graphs are used to display comparison at least two complementary numerical variables over an item/category. As we can see from the Figures 4, 6, and 8, the variables of interests, which are inpatient vs outpatient or inpatient vs surgeries performed, are closely related to each other, not directly mutually exclusive events. Furthermore, the offset (dodging) in the Figure 8 also confirms that columns are overlapped.

Lastly, the increasing trends in the number of inpatient treatments in Figures 6 and 8 may indicate that the hospital bed capacity increased over the years, whereas a stabile trend shows that the hospital's bed capacity did not change over the years. On the other hand, increasing trends in the number of outpatient treatments may indicate an increase in general service capacity of the hospital.

Grouped side-by-side bar graphs with two data series

Figure 10 represents the workload of sample hospitals between the years 1924 and 1937. There are also two groups here: inpatient and outpatient treatments, colored in black and white respectively. Figure 10 is different from previous ones since the columns for inpatient and outpatient treatments for the same year are now placed side-by-side accordingly. This can be done by setting the argument `position = position_dodge2()` in the `geom_col()` layer. The grouping variable is mapped into `fill` argument of the aesthetics of the main `ggplot()` call so that color of the levels of the grouping variable can be assigned manullay in `scale_fill_manual()` layer.

There is only one column in the year 1924 because there is no information available for the outpatient treatment in 1924. It is coded as **NA** in the data. The length of the column bars for the number of inpatient treatments are shorter than those of the outpatient treatments over the years. However, to increase the dominance of the number of inpatient treatments over the number of outpatient treatments, the inpatient treatment group is colored in black and placed ahead of outpatient treatment group, an information design stragety which we learned from White (1984). The data values for both groups are placed at the top of columns vertically due to enough space in the plotting area.

The Figures 6 and 10 are good examples that `geom_col()` layer in **ggplot2** can handle missing values in the data. In other words **ggplot2**can handle unequal length of data series for a given category (here it refers to a given year) while sketching overlapped and side-by-side bar graphs. On the other hand, a literature survey revealed that the Zonguldak Government Hospital in the Figure 6 and the sample hospitals in the Figure 10 were launched in 1924 as the second-stage hospitals in the cities offering inpatient services with specialized health workers such as doctors, nurses, and laboratory services. In the cities where these hospitals were available, a person with medical complications was initially admitted to the small hospitals with low health care capacity as the first-stage hospitals, and only the ones who needed inpatient services in a specialized medical area were transferred to the second-stage hospitals. For that reason, there is no data value for outpatient treatments in the Figures 6 and 10 in 1924. After 1924, due to on-going efforts improving health care policies and hospital capacities, these main hospitals started to offer both outpatient and inpatient treatments.

Figure 12 represents the laboratory workload for Malaria struggle between the years 1925 and 1937. The white color refers to the number of blood tests performed and the black color refers to the number of diagnoses. As in Figure 10, in Figure 12, the paired columns for the same year are now placed side-by-side. The columns for the number of blood tests are taller than the columns for the number of diagnoses over the years 1925 and 1937. Placing the number of blood tests ahead of the number of diagnoses enables us to compare how many blood tests resulted in a positive Malaria diagnosis (like today, the world is now comparing "the number of Covid-19 tests performed" with "the number of positive test results"). Although the number of diagnoses is smaller, coloring it in black increased its importance and softened the disparity between the sizes of the pair, which is an information design principle given in White (1984). Since the number of diagnoses falls behind the number of blood tests, vertically placing the horizontal axis tick mark labels in black color makes an illusion and increases the dominance of the number of diagnoses (in black) in the graphic.

The data values for the number of diagnoses are easily placed at the top of columns vertically due the shorter length of corresponding columns. However, while the data values for the number of blood tests are placed at the top of the column vertically between the years 1925 and 1928, the data values for the number of blood tests between the years 1929 and 1935 are placed inside the column bars vertically due to the space limitation in the plotting region.

In the Figures 10 and 12, the space between groups of bars is increased and now equals to the width of a single bar in a given group. On the other hand, space between bars within a group is not available.

The legends in the Figures 10 and 12 are placed vertically at the top-left of the figures since the overall structure of graphs are left-skewed. Note that the ordering in the group colors are also reflected in the legend keys as well. For example, in Figure 12, the first legend is in white and the second one is in black. Here we note that in the Figures 11 and 13, the legend key order is inherited from the order of

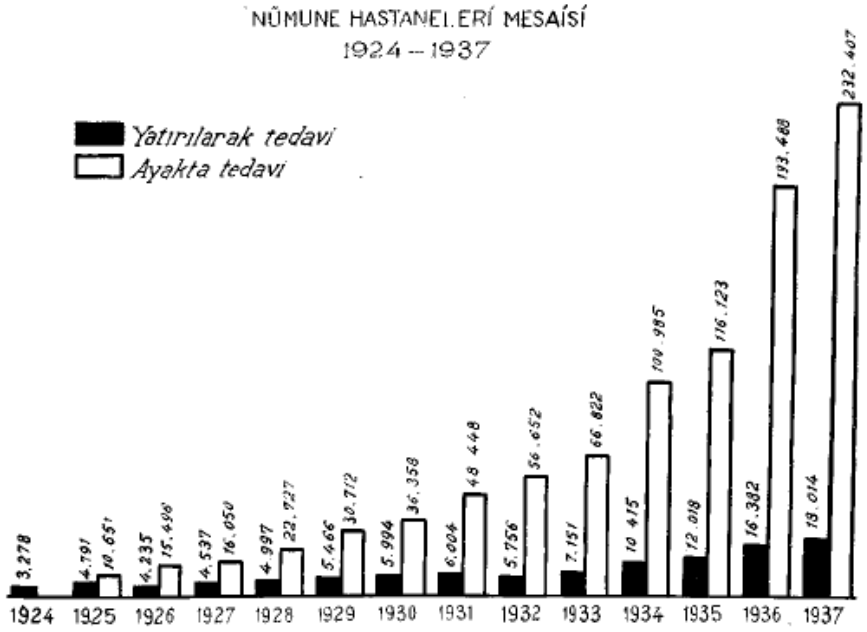


Figure 10: Historical figure on "The workload of sample hospitals, 1924-1937" retrieved from https://acikerisim.tbmm.gov.tr/handle/11543/553 (■ Inpatient treatment □ Outpatient treatment).

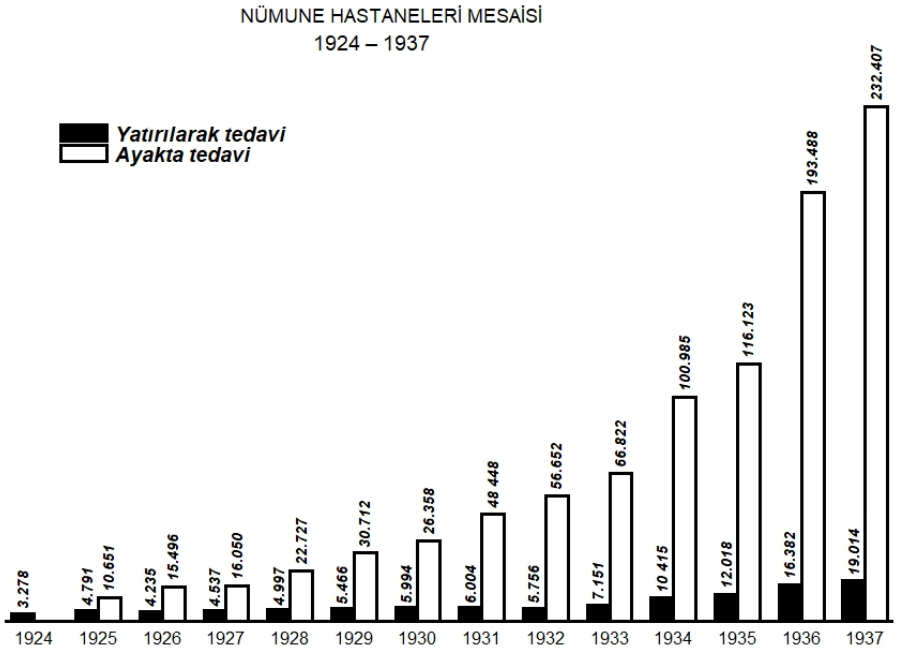


Figure 11: Reproduced figure on "The workload of sample hospitals, 1924-1937" (■ Inpatient treatment □ Outpatient treatment).

the level of the grouping variable in each plot and this grouping variable is declared in `fill` argument of the aesthetics of the main `ggplot()` call. The color of the legend keys are also matched with the color of the corresponding level of the grouping variable which was assigned in `scale_fill_manual` layer(). Lastly, from Figures 12, we can say that the government kept administrating blood tests over the years with an increasing trend and nearly 10% of them were being turned out to be positive.

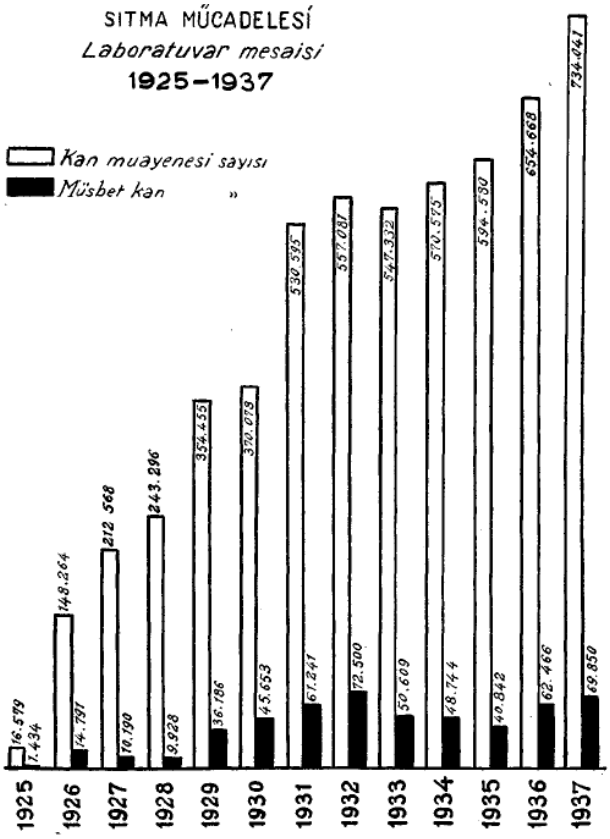


Figure 12: Historical figure on "The laboratory workload for Malaria struggle, 1925-1937" retrieved from https://acikerisim.tbmm.gov.tr/handle/11543/553 (□ The number of blood tests ■ The number of diagnoses).

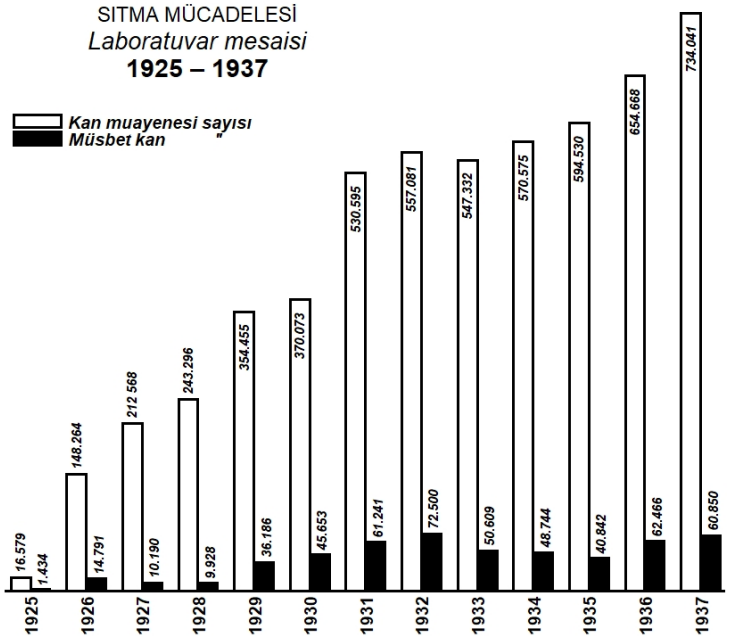


Figure 13: Reproduced figure on "The laboratory workload for Malaria struggle, 1925-1937" (□ The number of blood tests ■ The number of diagnoses).

Grouped side-by-side bar graph with more than two data series

Figure 14 represents the drugs sent by the Department of Control of Syphilis to cities for treatment between the period 1926 and 1937. Syphilis is a sexually transmitted disease caused by a bacteria and

it has bee a very serious public health problem in the world since 1800's. Several antibiotic-based drugs were used to cure the Syphilis.

In the Figure 14 we can see that there are four different drug types: Arsenobenzol, Bizmopen, Mercury, and Iodine, where they are filled in with black color, white color, textured with vertical lines, and textured with dots, respectively. This textured design is specifically called hatching in graphic design.

To emphasize grouping over the years, the horizontal axis is broken into segments. It can also be seen from the figure that there were Bizmopen and Mercury drugs in 1926 only; there were Arsenobenzol, Bizmopen, and Mercury drugs between the years 1927 and 1933, and afterwards, the fourth drug Iodine came out in 1934. Regardless of the number of drugs available at a specific year, the width of grouping is always kept uniform over the years 1926 and 1937.

The data series for the groups are placed side-by-side, which can be done by setting argument `position = position_dodge()` in the `geom_col()` layer. The order in the placement of the groups is also reflected in the legend keys. A side note is also attached to the graph telling that "The counts show kilo". Adding captions to the graphs is possible via `caption` argument in `labs()` layer of **ggplot2**.

On the other hand, we should say that some of the raw materials of these four drugs were imported from abroad, the treatment of syphilis was mandatory and free. Similarly, these four drugs were distributed free to the patients. In the Figure 14 the columns for the Mercury filled with vertical lines are very eye-catching due to their taller heights. While Wong (2016) says that Mercury had been commonly used to cure Syphilis until discovery of penicillin around 1940's , Mumyakmaz (2020) reported that the mercury was the most effective drug to treat the patients (95% effective), and the Iodine was the second effective drug.

The Figure 14 is a good example for use of hatching for differentiating the data groups when there is no color option or color printing was not easy or economical in 1938. However, this resulted in a challange that textured patterns are not allowed in the core functions of the **ggplot2** package. For that reason, we used several `annotate()` layers along with `segment` argument to integrate hatches with vertical lines and dots into the column bars. Since we did hatching manually, we could not synchronize textured patterns in the column bars with the legend keys. As a consequence of this, several `annotate()` layers with `rect` argument for drawing the legend boxes, several `annotate()` layers with `segment` argument for filling the textured patterns in legend keys, and several `annotate()` layers with `text` argument for typing the legend key titles were used.

We also provided three different historical column bar graphics along with `R` codes in the Supplementary material for further interest.

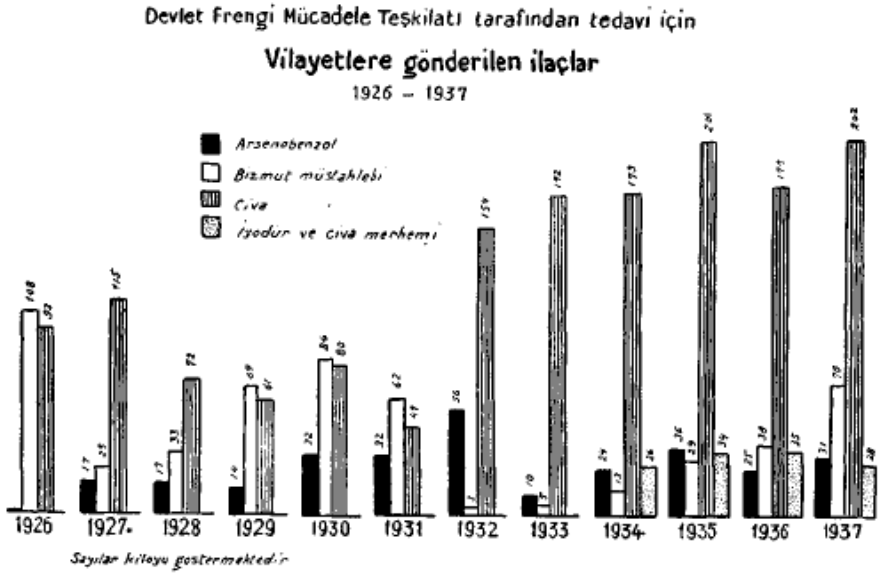


Figure 14: Historical figure on "The drugs sent by the Department of Control of Syphilis to cities for treatment, 1926-1937" retrieved from https://acikerisim.tbmm.gov.tr/handle/11543/553 (■ Arsenobenzol □ Bizmopen ▩ (textured pattern with vertical lines) Mercury ⁝⁝⁝ Iodine).

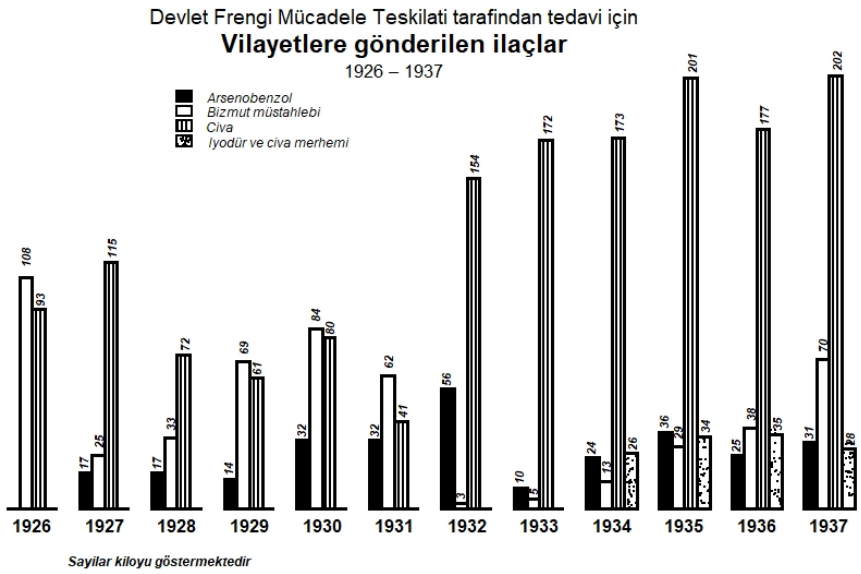


Figure 15: Reproduced figure on "The drugs sent by the Department of Control of Syphilis to cities for treatment, 1926-1937" (■ Arsenobenzol □ Bizmopen ▒ (textured pattern with vertical lines) Mercury ⁝⁝ Iodine).

Paired bar graphs with two data series

Figure 16 shows the total number of serums and vaccines, which are produced and consigned, at the Central Hygiene Institute for the years between 1930 and 1937. The Central Hygiene Institute was a reference laboratory in the country launched for producing serum and vaccine for several infectious diseases such as rabies, typhoid, and whooping cough.

Figure 16 breaks down the data into two panels as produced (left-panel) and cosigned (right-panel) through a vertical axis. The left side of vertical axis is for the number of produced serum and vaccine items in kilogram and the right side of the vertical axis is for the number of consigned serum and vaccine items in kilogram. Under each panel, the data set for serum (in black) and vaccine (in white) are displayed via overlapping columns from 1930 to 1937. The legend keys for serum (in black) and vaccine (in white) are separated horizontally. Note that this is the first time a vertical axis appears in a graph and it is placed at the center of the graph. It denotes the years with the title "Yillar" and descends from 1937 to 1930. Since the columns of recent years are taller than the columns of earlier years, which also shows the continuing success of the institute, reversing the order of years (starting from the recent year 1937) makes the graphic to look like a population pyramid. Furthermore, although data values are annotated to the column bars, a horizontal axis with tick marks and tick mark labels are also available. The tick mark labels show the respective counts, which are not linear in magnitude, going from 0 to 4500 with unequal increments. The horizontal axis title "SAYISI" refers to the "Frequency". This is the first time we have a horizontal axis title as well. The text aligned with 45 degree at the left panels refers to the "PRODUCED (IN KILO)" and the one with 135 degree at the right refers to the "CONSIGNED IN (KILO)", which are symmetric to each other with respect to the vertical axis. We can consider this graph as a paired bar graph having a double horizontal axis with a common vertical axis. Lastly, here we can see that the country was very successful both at producing and consigning vaccine in 1930's due to increasing trend over the years.

Due to high infant mortality rates in the early years of the country, special effort was devetod the health care of mother and child. Figure 18 shows the service of birth and childcare houses for women and children between the period 1926 and 1937. As in Figure 16, the vertical axis breaks down the data into two panels as inpatient services (left-panel) and outpatient services (right-panel). Then for each panel, the data set for child and woman are displayed over the years 1926 and 1937 as side-by-side column bars in black and white colors, respectively. As in Figure 16, the vertical axis at the center denotes years by "Yillar" starting from 1937 and descending to 1926. The tick marks on horizontal axis show the respective frequencies with linear increments. The tick mark labels at left side of the horizontal axis represents the numbers in terms of "Thousands", going from 0, then to 1, 2, 3, 4, and 5000. Similarly, tick mark labels at the right side of the horizontal axis represents the numbers in terms of "Ten Thousands", going from 0, then to 10, 20, 30, 40, and 50,000. Due to space limitation on the horizontal axis, the last three digits of the numbers are not used, except the last numbers (5,000 at the

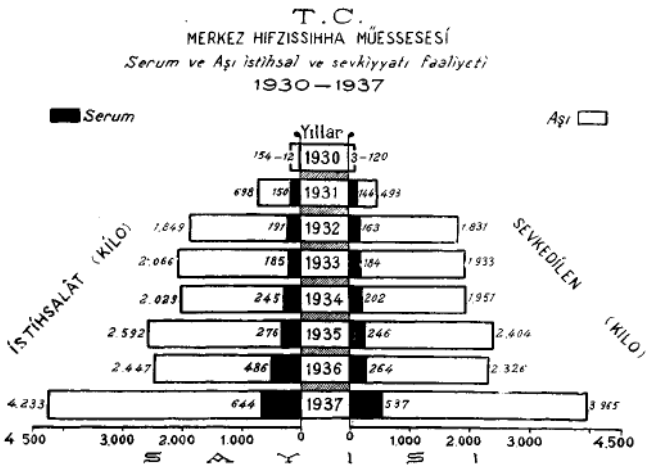


Figure 16: Historical figure on "The service of produced and consigned serum and vaccine in kilogram at the Central Hygiene Institute, 1930-1937" retrieved from https://acikerisim.tbmm.gov.tr/handle/11543/553 (The left-panel is for produced items and the right-panel is for consigned times; ■ Serum □ Vaccine).

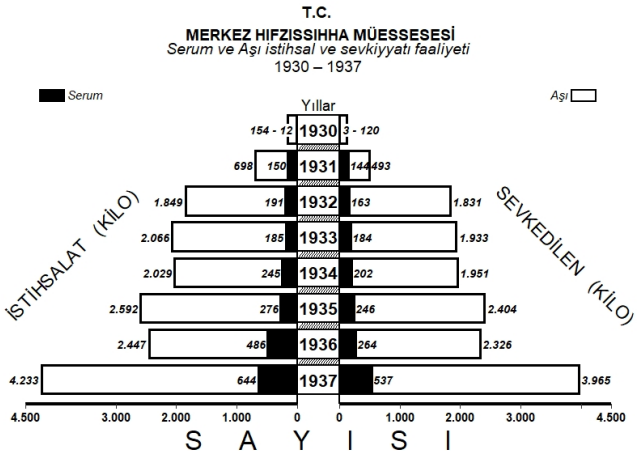


Figure 17: Reproduced figure on "The service of produced and consigned serum and vaccine in kilogram at the Central Hygiene Institute, 1930-1937" (The left-panel is for produced items and the right-panel is for consigned times; ■ Serum □ Vaccine).

left and 50,000 at the right). This looks like a promising example for information design of displaying very large numbers on graphics when there is no space to integrate all the text into the plotting area. Integration of these tick mark labels is done via `labels` argument of `scale_y_continuous()` layer in **ggplot2**. The paired bar graphs in the Figures 17 and 19 can be plotted via using multiple `geom_col()` layers with some extra aesthetic work in **ggplot2** package. Here we should note that to be able to assign a pyramid look to the graph, the data is visualized with an illusion since height of the left-panel bars are actually pretty much smaller than the ones at right-panel. Lastly, we can say that the country was also succesful at taking care of childs and mothers with an increasing number of inpatient and outpatient services over the years.

4 Redesign of historical bar graphics in modern era

The historical graphics given above enable us to investigate the trends of several numerical variables over time and make comparisons between these numerical variables through column bar graphics. We can also re-visualize these graphics with the help of modern data visualization principles and software technology to increase the readibility and effectiveness of the graphs through increasing the

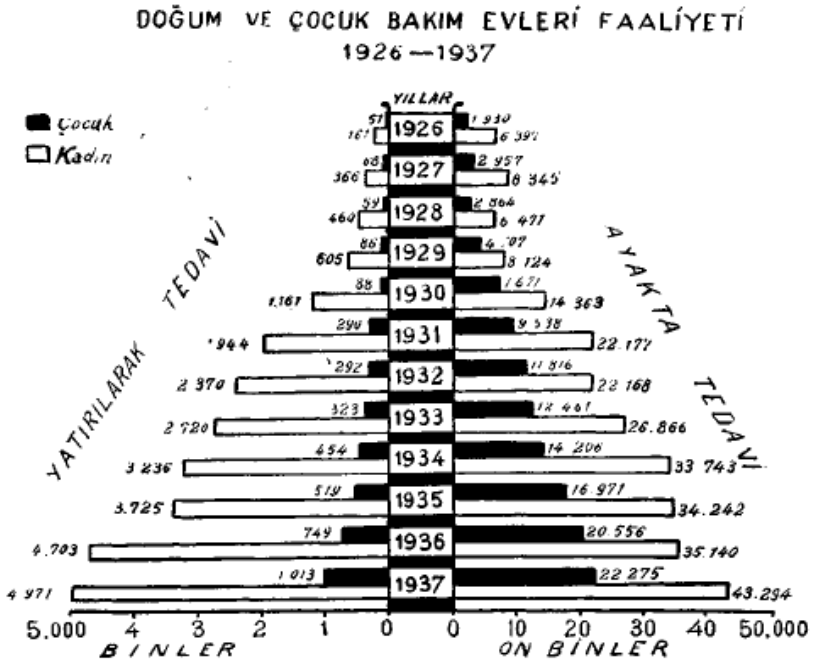


Figure 18: Historical figure on "The service of birth and childcare houses, 1926-1937" retrieved from https://acikerisim.tbmm.gov.tr/handle/11543/553 (The left-panel is for inpatient services and the right-panel is for outpatient services; ■ Child □ Woman).

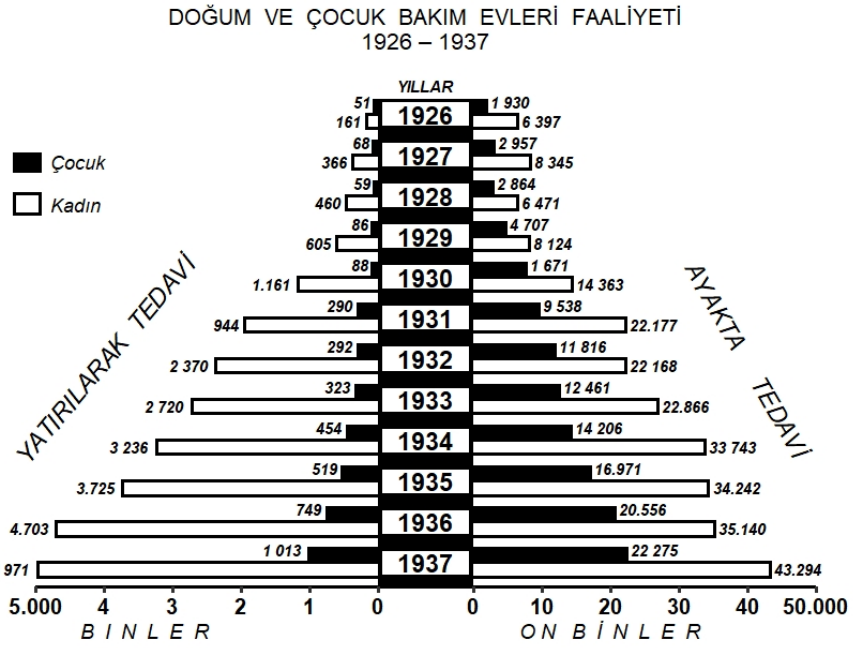


Figure 19: Reproduced figure on "The service of birth and childcare houses, 1926-1937" (The left-panel is for inpatient services and the right-panel is for outpatient services; ■ Child □ Woman).

data-ink ratio given below:

$$\text{Data-ink ratio} = \frac{\text{Ink used to describe the data}}{\text{Ink used to visualize graph}}.$$

The higher the ratio, the better the visualization comes out. Since the nature of the data in the historical graphics given in this paper is time series, line grahs can also be alternatively used to visualize the same data with a higher data-ink ratio. For example, as we discussed in Figure 6, the pattern of the number of inpatient treatments and the number of outpatient treatments over time cannot be detected easily and requires some cognitive effort due to the data structure and overlapping column bar design. In Figure 20, we used a multi-line plot to display the number of patients treatments and that of outpatient treatments between 1924 and 1937 with `geom_line()` layer. A regular line is now used to represent the number of inpatient treatments, whereas a dashed line is preferred for the number of outpatient treatments. Color is not assigned to lines so that the graphic is visually more accessible. Unlike Figure 6, a vertical axis along with axis labels with linear increments from 0 to 5000 is used to quantify the frequencies. The last data values of two data group are placed on the figure only. The legend is removed from the figure and data group categories are annotated next to the line of interest to decrease the ink used to identify non-data values. Unlike Figure 6, in Figure 20, it is now more clearly seen that the number of outpatient services is not available in the year 1924, both data groups have an increasing trend over the years, and the difference between quantities of the number of inpatient treatments and that of outpatient treatments is non-negative until the year 1932, then it is negative onwards. Furthermore, reducing the overall ink used to draw the graphic also results in a decrease in computational burden to implement this graphic. The amount of lines required to implement new figure decreased from 48 lines (2297 characters) to 26 lines (1091 characters) (please have a look at the `R` codes in the Supplementary to implement Figures 7 and 20).

With the **plotly** package, Figure 20 can be further turned into an interactive graphic for web-based publications, where the data values and other components of the graphic are interacted with mouse-over and can be removed and added back with mouse clicks. However, interactive graphs cannot be feasible for hard-copy prints.

Figure 20: An alternative illustration of "The service of the Zonguldak Government Hospital, 1924-1937" (——— The number of inpatient treatments - - - - - - The number of outpatient treatments).

Another example can be the Figure 14, where the amount of four different drugs, namely, Arsenobenzol, Bizmopen, Mercury, and Iodine, sent by the Department of Control of Syphilis to cities for treatment is displayed. In Figure 14, it is easy to compare the amount of four different drugs to each other for a given year. This design choice eliminates the ability to investigate trends in the amount of a specific drug supplied over the years. To investigate the relationship between and within the drug categories, we can prefer displaying the amount of each drug over the years separately through faceting with `facet_wrap()` layer.

Figure 21 gives the amount of Arsenobenzol, Bizmopen, Mercury, and Iodine, sent by the Department of Control of Syphilis as a facet line plot, where each sub-panel refers to an individual drug category. We can differentiate the drug categories via stripe titles. Each sub-panel now sits on a common horizontal axis, that's years from 1925 to 1937, and a common vertical axis changing from to 0 to the largest possible value in the overall data. Hence, we can clearly and fairly investigate the trend of each drug over the years, and compare drug amounts for a given year. Faceting also enables us to avoid hatching and using legends, resulting in a decrease in the computational burden to implement this figure such that the amount of lines required to code new figure decreased from 101 lines (4520 characters) to 34 lines (1441 characters) (please have a look at the `R` codes in the Supplementary material to implement Figures 15 and 21).

Lastly, Figure 18 shows the service of birth and childcare houses between the years 1926 and 1937.

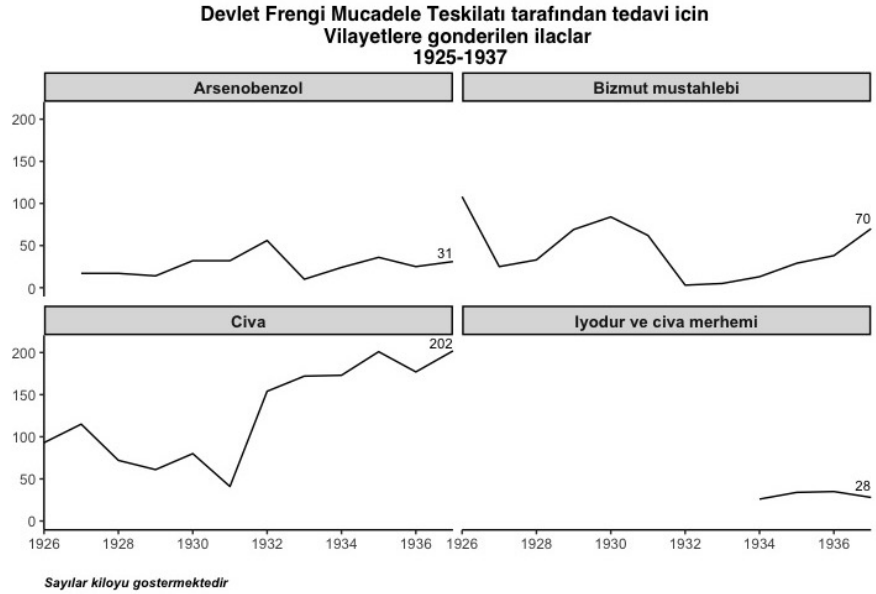


Figure 21: An alternative illustration of "The drugs sent by the Department of Control of Syphilis to cities for treatment, 1926-1937" (The upper-left panel is Arsenobenzol, the upper-right panel is Bizmopen, the lower-left panel is Mercury, and the lower-rightpanel is Iodine).

The inpatient and the outpatient services are further divided into two categories: Child and Woman. The original design of the graph resembles a population pyramid with the help of creating an illusion that the left-panel (for inpatient services) and the right-panel (for outpatient services) are symmetric to each other over the vertical axis, when they are not. Indeed, while the left horizontal axis spans from 0 to 5000, the right horizontal axis from 0 to 50000 with linear increments. Eliminating 0 strings from horizontal axis labels also contributes this confusion. In the end, as we discussed earlier, this perceptional illusion results in an aesthetically pleasing, but misleading graph. On the other hand, since left and right panels do not share a common horizontal axis, faceting will not result in a fair comparison between panels as done in Figure 21. Alternatively, Figure 18 can be split into two sub-line plots with the same horizontal axis, that's the years from 1926 to 1937, and with different vertical axis giving the relative frequencies for inpatient services and outpatient services, respectively. This leads to the Figure 22 which gives more realistic comparison between child and women in terms of inpatient and outpatient services received. Lastly, the amount of lines required to code new figure decreased from 80 lines (3578 characters) to 55 lines (2150 characters) (please have a look at the `R` codes in the supplementary material to implement Figures 19 and 22).

For a detailed discussion on effectiveness of graphics, chart design, perception and cognition, we kindly invite readers to read Cleveland and McGill (1986) and Vanderplas et al. (2020).

5 Conclusion

In this study, our aim was two-fold: first understanding the information design behind the historical column bar graphics drawn with hand and published in late 1930's, and then, if possible, reproducing these graphics with the help of the advances in data visualization technologies in our era, namely, through **ggplot2** package.

While we were dealing with these historical graphics to reproduce them in **ggplot2** package, we were mostly challanged with i) multi-line titles with different font styles, ii) textured patterns, and iii) data groups where the difference of the frequencies is not monotonic over the years.

In a multi-line title or any multi-line text within a figure plotting area such as tick mark labels, data labels, legends, and so on, if interest is on changing font face i.e., making text bold or italic, then simple `Markdown` syntax would be integrated into text and rendered with the help of `element_markdown()` layer in the **ggtext** package (Wilke, 2020). However, if more aesthetic changes such as font family type, size, or color are needed in the text, then the text can be manipulated appropriately with the corresponding `HTML` tags, and rendered with `element_textbox()` layer in the **ggtext** package. We also provided `R` codes to produce the multi-lines in the Figure 2 and 14 in the supplementary material.

On the other hand, **ggpattern** package (FC, 2020) provides geometric based patterns such as stripe,

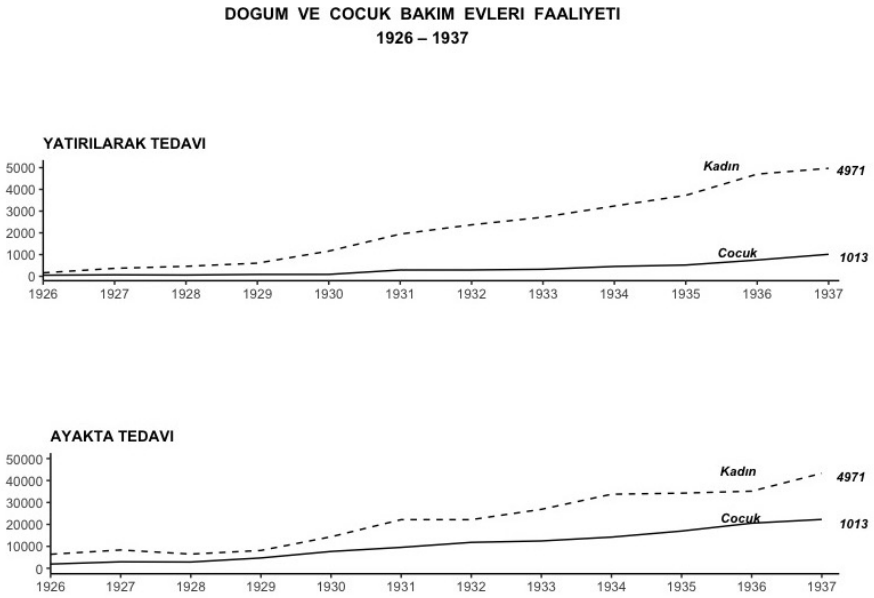


Figure 22: An alternative illustration of "The service of birth and childcare houses, 1926-1937" (The upper-panel is for inpatient services and the right-panel is for outpatient services; ——— Child - - - - - - Woman).

crosshatch, or circle to `ggplot2` objects with `geom_col_pattern()` layer. If a specific pattern is required in bars, then `geom_pattern_manual()` layer enables to assign the desired pattern to a specific bar. This order is reflected into the legend keys as well. For comparison, we reproduced the Figure 15 with **ggpattern** and presented it in Figure 23. The amount of lines to produce Figure 23 is now 51 (2171 characters), where `R` codes are available in the supplementary material.

Unfortunately, the last challenge requires more work which can be considered as a future study.

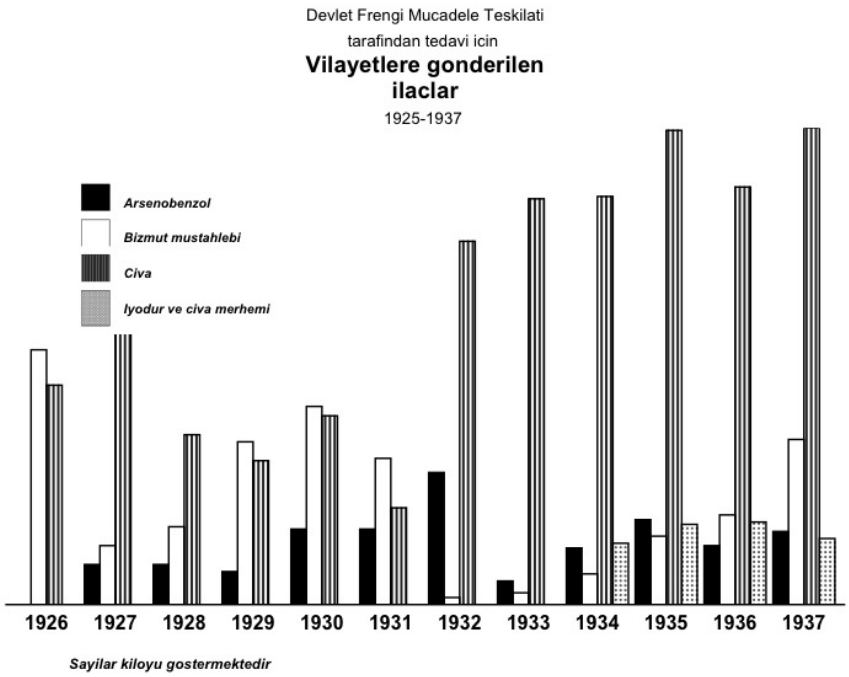


Figure 23: An alternative illustration of "The drugs sent by the Department of Control of Syphilis to cities for treatment, 1926-1937" (■ Arsenobenzol □ Bizmopen ▩ (textured pattern with vertical lines) Mercury ⁚ Iodine).

In today's Covid-19 pandemic, we also saw that data visualization helped us to better understand

the Covid-19 related statistics, i.e., the number of confirmed cases, the number of recovered cases, the number of active cases, and the number of deaths. It can be said that John Burn-Murdoch's Financial Times charts played a leading role in the visualization of Covid-19 related statistics through line charts. With the help of today's technological advances in data visualization, many other media outlets such as New York Times and the Guardian take advantage of zoomable and scrollable graphics for visually attractive story telling. Furthermore, unlike the past, GIS-based interactive data visualization examples such as CNN health's Covid-19 tracker (Wolfe et al., 2021) came into play for spatially investigating the progress of the disease and/or vaccination. Nevertheless, understanding all these visualizations from the viewer's side requires data literacy.

What we experienced during the Covid-19 pandemic also enabled us to better understand the historical graphics used in our study. When we were dealing with these graphics back in late 2019, it was our limited understanding of how important the workload and capacity of hospitals was during an epidemic or a pandemic and how important the services of government, private, and mobile hospitals were for carrying the workload in the fight against the infectious diseases in order to "flatten the curve". Furthermore, we also learned that, as in the Figure 13, while epidemics take a very long time to be diminished from the world and even if the ratio of "the number of blood tests" to "the number of positive test results" is getting larger over the years, increasing the test capacity was also an old school approach yielding the idea "The more tests the better prevention is".

Lastly, we can conclude that neither pandemics, nor the data visualization is new to our world. As in the past and today, statistical graphics and data visualization play a vital bridge role between the authorities and the public during global issues such as health.

Acknowledgment

We would like to thank Assoc. Prof. Dr. Eminalp Malkoc from Department of History at Istanbul Technical University for providing us the original statistical graphics used in this paper. We would also like to thank the reviewer whose comments improved the quality of the paper.

Bibliography

J. R. Beniger and D. L. Robyn. Quantitative graphics in statistics: A brief history. *The American Statistician*, 32(1):1–11, 1978. [p147]

W. S. Cleveland and R. McGill. An experiment in graphical perception. *International Journal of Man-Machine Studies*, 25(5):491–500, 1986. [p163]

P. del Consiglio dei Ministri. Dati COVID-19 Italia, 2020. URL https://github.com/pcm-dpc/COVID-19. [p146]

O. Durmaz. Speak to the eyes: Visualizing information from the Ottoman era to the Republic, 2017. URL https://www.arch.columbia.edu/exhibitions/61-speak-to-the-eyes. [p146]

M. FC. *ggpattern: Geoms with patterns for ggplot2.*, 2020. URL https://coolbutuseless.github.io/package/ggpattern. R package version 0.2.0. [p163]

M. Friendly. The golden age of statistical graphics. *Statistical Science*, 23(4):502–535, 2008. [p146]

A. Garthwaite. Visualising the pandemic: Interviews with data journalists covering Covid-19, 2020. URL https://www.significancemagazine.com/science/655-visualising-the-pandemic-interviews-with-data-journalists-covering-covid-19?fbclid=IwAR2bcEMmxBv-Oyr7bqTuVWQKbL-hgKuVsEKD0WBA9T3IcDTL2Gj_ARip6Vo. [p146]

A. Gelman and A. Unwin. Infovis and statistical graphics: Different goals, different looks. *Journal of Computational and Graphical Statistics*, 22(1):2–28, 2013. [p146]

R. Harris. *Information graphics: A comprehensive illustrated reference*. Oxford University Press, 2000. [p147]

Johns Hopkins University. Covid-19 dashboard by the Center for Systems Science and Engineering, 2020. URL https://gisanddata.maps.arcgis.com/apps/opsdashboard/. [p146]

E. Malkoc. Fight against syphilis that forgot to embrace in the era of early Republic (in Turkish), 2018. URL https://www.tarihvakfi.org.tr/Images/UserFiles/Documents/Gallery/01-03-icindekiler-2.pdf. [p146]

A. Matthew. ggplot2 meets W. E. B. Du Bois, 2019. URL https://www.statswithmatt.com/post/ggplot2-meets-w-e-b-du-bois/#fnref1. [p146]

E. McCoy. For public health organizations, data visualization is not a choice, 2020. URL https://medium.com/nightingale/for-public-health-organizations-data-visualization-is-not-a-choice-e945f54152b8. [p146]

H. G. Mumyakmaz. Terrific disease: Struggling with syphilis in the first half of the 20th century in turkey (in turkish). *Akademik Hassasiyetler*, 7(13):119–148, 2020. [p158]

A. Quito and D. Kopf. Designers and statisticians disagree on what makes a good information graphic, 2020. URL https://qz.com/1781388/criteria-for-good-data-visualization-according-to-design-and-statistics. [p146]

B. Sengul. The voyage of information visualization in history (in Turkish), 2017. URL https://yazname.com/bilgi-gorsellestirmenin-tarihteki-yolculugu/24/06/2017. [p146]

Toplumsal Tarih. Pandemics in history (in Turkish), 2018. URL https://www.tarihvakfi.org.tr/dergiler/toplumsal-tarih/296/toplumsal-tarih-agustos2018/5. [p146]

S. VanderPlas, G. Ryan, and H. Hofmann. Framed! Reproducing and revisiting 150-year-old charts. *Journal of Computational and Graphical Statistics*, 28(3):620–634, 2019. [p146]

S. Vanderplas, D. Cook, and H. Hofmann. Testing statistical charts: What makes a good graph? *Annual Review of Statistics and Its Application*, 7:61–88, 2020. [p163]

J. White. *Using charts and graphs: One thousand ideas for getting attention using charts and graphs*. 1984. [p149, 155]

H. Wickham, W. Chang, L. Henry, T. L. Pedersen, K. Takahashi, C. Wilke, K. Woo, H. Yutani, and D. Dunnington. *ggplot2: Ceate Elegant Data Visualisations Using the Grammar of Graphics*, 2020. URL https://CRAN.R-project.org/package=ggplot2. R package version 3.3.0. [p147]

C. O. Wilke. *ggtext: Improved Text Rendering Support for 'ggplot2'*, 2020. URL https://cran.r-project.org/web/packages/ggtext. R package version 0.1.1. [p163]

D. Wolfe, B. Manley, and P. Krishnakumar. Tracking COVID-19 vaccines in the US, 2021. URL https://edition.cnn.com/interactive/2021/health/us-covid-vaccinations/. [p165]

S. Wong. Syphilis and the use of mercury, 2016. URL https://www.pharmaceutical-journal.com/opinion/blogs/syphilis-and-the-use-of-mercury/20201679.blog. [p158]

Sami Aldag
Department of Mathematics Engineering,
Istanbul Technical University,
Istanbul, 34469,
Turkey
aldag@itu.edu.tr

Dogukan Topcuoglu
Department of Mathematics Engineering,
Istanbul Technical University,
Istanbul, 34469,
Turkey
topcuoglud@itu.edu.tr

Gul Inan
Department of Mathematics,
Istanbul Technical University,
Istanbul, 34469,
Turkey
(ORCiD: 0000-0002-3981-9211)
inan@itu.edu.tr

spherepc: An R Package for Dimension Reduction on a Sphere

by Jongmin Lee, Jang-Hyun Kim and Hee-Seok Oh

Abstract Dimension reduction is a technique that can compress given data and reduce noise. Recently, a dimension reduction technique on spheres, called spherical principal curves (SPC), has been proposed. SPC fits a curve that passes through the middle of data with a stationary property on spheres. In addition, a study of local principal geodesics (LPG) is considered to identify the complex structure of data. Through the description and implementation of various examples, this paper introduces an R package **spherepc** for dimension reduction of data lying on a sphere, including existing methods, SPC and LPG.

1 Introduction

This paper aims to introduce an R package **spherepc** that considers several dimension reduction techniques on a sphere, which encompass recently developed approaches such as SPC and LPG as well as some existing methods, and discuss how to implement these methods through **spherepc**.

Dimension reduction methods are widely used in various fields, including statistics and machine learning, by efficiently compressing data and removing noise (Benner et al., 2005). As one of the dimension reduction methods, the principal curves of Hastie and Stuetzle (1989) are suitable for fitting a curve or a surface of data in Euclidean space, which go through the middle of the data. Hauberg (2016) proposed an algorithm to find the principal curves in Riemannian manifolds based on the concept of the original principal curves. However, the principal curves proposed by Hauberg (2016) no longer represent the data continuously because of the approximation of the projection step required to fit the curves.

Recently, Lee et al. (2021a) proposed a new method, termed spherical principal curves (SPC), that constructs principal curves, ensuring a stationary property on spheres. SPC is useful for representing circular or waveform data with smaller reconstruction errors than conventional methods, including principal geodesic analysis (Fletcher et al., 2004), exact principal circle (Lee et al., 2021a), and principal curves proposed by Hauberg (2016). However, SPC has the disadvantage of being sensitive to initialization. As a result, there are some data structures that SPC does not apply to, for example, data with spirals, zigzags, or branches like tree-shape. A localized version of SPC called local principal geodesics (LPG) is being developed to resolve such a problem. A function for LPG is also provided in the package **spherepc**. Research on the LPG is underway in progress.

To the best of our knowledge, no available R packages offer the methods of dimension reduction and principal curves on a sphere. The existing R packages providing principal curves, such as **princurve** (Hastie and Weingessel, 2015) and **LPCM** (Einbeck et al., 2015), are available only on Euclidean space, not on a sphere or (Riemannian) manifold. In addition, most dimension reduction methods on manifolds (Huckemann et al., 2010; Panaretos et al., 2014; Liu et al., 2017) involve somewhat complex optimizations. The proposed package **spherepc** for R provides the state-of-the-art principal curve technique on the sphere (Lee et al., 2021a) and comprehensively collects and implements the existing methods (Fletcher et al., 2004; Hauberg, 2016).

The rest of this paper is organized as follows. The following section introduces the existing methods for dimension reduction on the sphere and relevant functions covered in the package **spherepc**, which is available on CRAN. Furthermore, their usages are discussed with examples in detail. Then, the spherical principal curves proposed by Lee et al. (2021a) and principal curves of Hauberg (2016) are briefly described. In addition, implementations of the `SPC()` and `SPC.Hauberg()` functions in the **spherepc** are presented. The subsequent section discusses the local principal geodesics (LPG) with the implementation of various simulated data, demonstrating its promising usability. In the application session, all the mentioned methods are performed to analyze real seismological data. Finally, conclusions are given in the last section.

2 Existing methods

Principal geodesic analysis

Principal geodesic analysis (PGA) proposed by Fletcher et al. (2004) can be regarded as a generalization of principal component analysis (PCA) to Riemannian manifolds. In particular, Fletcher et al. (2004)

performed dimension reduction of data on the Cartesian product space of the manifolds. In detail, the data are projected onto the tangent spaces at the intrinsic means of each component of the manifolds; thus, the given data are approximated as points on Euclidean vector space, and subsequently, PCA is applied to the points. As a result, the dimension reduction can be performed through the inverse of the tangent projections.

The principal geodesic analysis can be implemented by the `PGA()` function available in the **spherepc**. The detailed usage of the `PGA()` function is described as follows.

```
PGA(data, col1 = "blue", col2 = "red")
```

Before using the `PGA()` function, it requires loading the packages **rgl** (Adler and Murdoch, 2020), **sphereplot** (Robotham, 2013), and **geosphere** (Hijmans et al., 2017). The following codes yield an implementation of the `PGA()` function.

```
#### for all simulated datasets, longitude and latitude are expressed in degrees
#### example 1: half-great circle data
> circle <- GenerateCircle(c(150, 60), radius = pi/2, T = 1000)
> sigma <- 2                                # noise level
> half.circle <- circle[circle[, 1] < 0, , drop = FALSE]
> half.circle <- half.circle + sigma * rnorm(nrow(half.circle))
> PGA(half.circle)

#### example 2: S-shaped data
# the dataset consists of two parts: lon ~ Uniform[0, 20],
# lat = sqrt(20 * lon - lon^2) + N(0, sigma^2),
# lon ~ Uniform[-20, 0], lat = -sqrt(-20 * lon - lon^2) + N(0, sigma^2)
> n <- 500
> sigma <- 1                                # noise level
> lon <- 60 * runif(n)
> lat <- (60 * lon - lon^2)^(1/2) + sigma * rnorm(n)
> simul.S1 <- cbind(lon, lat)
> lon2 <- -60 * runif(n)
> lat2 <- -(-60 * lon2 - lon2^2)^(1/2) + sigma * rnorm(n)
> simul.S2 <- cbind(lon2, lat2)
> simul.S <- rbind(simul.S1, simul.S2)
> PGA(simul.S)
```

Because a principal geodesic is always a great circle, the `PGA()` function is suitable for identifying the global data trend. The implementations of half-circle and S-shaped data are displayed in Figure 1, where the principal geodesic properly extracts the global trends in the half-great circle and S-shaped data, while it cannot identify the circular variations in the S-shaped case. In addition, the arguments and outputs of the `PGA()` function are described in Tables 1 and 2.

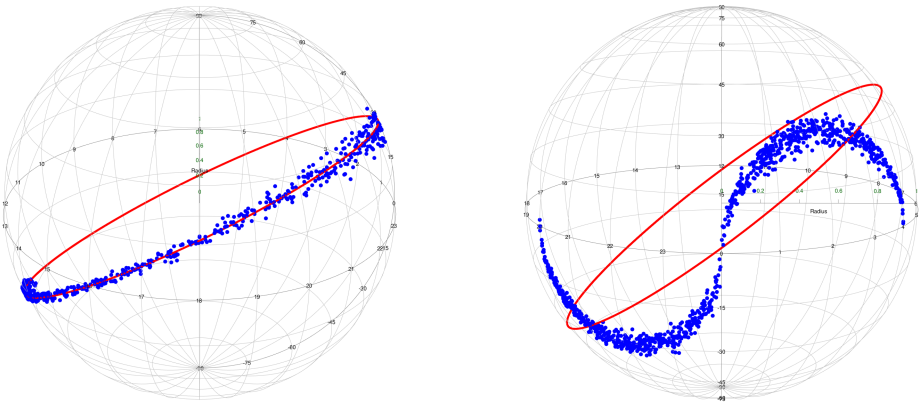


Figure 1: From left to right, half-great circle and S-shaped data (blue) and the results (red) of principal geodesic analysis (PGA). The principal geodesic detects the global trends of the noisy half-great circle and the S-shaped data but cannot identify the circular variation of the S-shaped data.

Argument	Description
`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude (denoted by degrees).
`col1`	color of data. The default is blue.
`col2`	color of the principal geodesic line. The default is red.

Table 1: Arguments of the `PGA()`.

Output	Description
`plot`	plotting of the result in 3D graphics.
`line`	spatial locations (longitude and latitude by degrees) of points in the principal geodesic line.

Table 2: Outputs of the `PGA()`.

Principal circle

In a spherical surface, as shown in Figure 1, the principal geodesic analysis always results in a great circle, which cannot be sufficient to identify the non-geodesic structure of data. The circle on a sphere that minimizes a reconstruction error is called a principal circle, where the reconstruction error is defined as the total sum of squares of geodesic distances between the circle and data points. However, the existing method for generating the principal circle is still based on the tangent space approximation and its inverse process, thereby leading to numerical errors. Lee et al. (2021a) have proposed an exact principal circle in an intrinsic way and its practical algorithm based on gradient descent. The details are described in Section 3 of Kim et al. (2020) and Appendix B of Lee et al. (2021b). The **spherepc** package provides the `PrincipalCircle()` function to implement the intrinsic principal circle. Its usage is followed by

```
PrincipalCircle(data, step.size = 1e-3, thres = 1e-5, maxit = 10000).
```

Argument	Description
`data`	matrix or data frame consisting of spatial locations (longitude and latitude denoted by degrees) with two columns.
`step.size`	step size of gradient descent algorithm. For convergence of the algorithm, `step.size` is recommended to be below 0.01. The default is 1e-3.
`thres`	threshold of the stopping condition. The default is 1e-5.
`maxit`	maximum number of iterations. The default is 10000.

Table 3: Arguments of the `PrincipalCircle()`.

The arguments of the `PrincipalCircle()` are described in Table 3, and its output is a three-dimensional vector, where the first and second components are longitude and latitude (represented by degrees), respectively. The last one is the radius of the principal circle. To display the circle, the `GenerateCircle()` function should be implemented. Its usage is followed by

```
GenerateCircle(center, radius, T = 1000).
```

The output of the `GenerateCircle()` function is a matrix consisting of spatial locations (longitude and latitude by degrees) with two columns, which can be plotted by the `sphereplot::rgl.sphgrid()` and `sphereplot::rgl.sphpoints()` functions from the **sphereplot** package (Robotham, 2013). Note that the **sphereplot** package depends on the **rgl** package (Adler and Murdoch, 2020). The detailed arguments of the `GenerateCircle()` function are described in Table 4.

The following codes implement principal circles by the `PrincipalCircle()` and `GenerateCircle()` functions.

```
## for all the following examples, longitude and latitude are denoted by degrees
#### example 1: half-great circle data
> circle <- GenerateCircle(c(150, 60), radius = pi/2, T = 1000)
> half.great.circle <- circle[circle[, 1] < 0, , drop = FALSE]
> sigma <- 2                          # noise level
```

Argument	Description
`center`	center of circle with spatial locations (longitude and latitude denoted by degrees).
`radius`	radius of circle. It should be range from 0 to π.
`T`	the number of points that make up a circle. The points in a circle are equally spaced. The default is 1000.

Table 4: Arguments of the `GenerateCircle()`.

```
> half.great.circle <- half.great.circle + sigma * rnorm(nrow(half.great.circle))
## find a principal circle
> PC <- PrincipalCircle(half.great.circle)
> result <- GenerateCircle(PC[1:2], PC[3], T = 1000)
## plot the half-great circle data and principal circle
> sphereplot::rgl.sphgrid(col.lat = "black", col.long = "black")
> sphereplot::rgl.sphpoints(half.great.circle, radius = 1, col = "blue", size = 9)
> sphereplot::rgl.sphpoints(result, radius = 1, col = "red", size = 6)

#### example 2: circular data
> n <- 700                              # the number of samples
> sigma <- 5                            # noise level
> x <- seq(-180, 180, length.out = n)
> y <- 45 + sigma * rnorm(n)
> simul.circle <- cbind(x, y)
## find a principal circle
> PC <- PrincipalCircle(simul.circle)
> result <- GenerateCircle(PC[1:2], PC[3], T = 1000)
## plot the circular data and principal circle
> sphereplot::rgl.sphgrid(col.lat = "black", col.long = "black")
> sphereplot::rgl.sphpoints(simul.circle, radius = 1, col = "blue", size = 9)
> sphereplot::rgl.sphpoints(result, radius = 1, col = "red", size = 6)
```

The results of the principal circle are shown in Figure 2. As one can see, the principal circle identifies the circular patterns of the noisy half-great circle and circular dataset well.

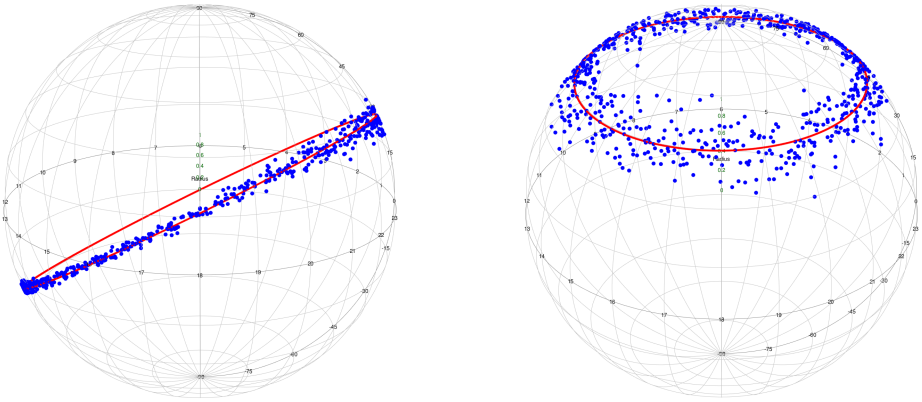


Figure 2: Half-great circle data and circular data (blue) and the results (red) of the principal circle from left to right. The principal circle can identify the relatively small circular structure (right) and the great circle structure (left).

3 Spherical principal curves

Principal curves proposed by Hastie and Stuetzle (1989) can be considered as a nonlinear generalization of the principal component analysis in the sense that the principal curves pass through the middle of given data and reserve a stationary property. The curve is a smooth function from a one-dimensional closed interval to a given space; then, a curve f is said to be a principal curve of X or self-consistent if

the curve satisfies

$$f(\lambda) = \mathbb{E}\big(X \mid \lambda_f(X) = \lambda\big),$$

where $f(\lambda_f(x))$ is the closest (projection) point in the curve f from the point x.

Hauberg (2016) provided an algorithm for principal curves on Riemannian manifolds. However, Hauberg (2016) used approximations for finding the closest point of each data point, which may lead to numerical errors. Recently, Lee et al. (2021a) presented theoretical results of principal curves on spheres and a practical algorithm for constructing principal curves without any approximations, called spherical principal curves (SPC), thereby causing the given data to be represented more precisely and smoothly compared to principal curves of Hauberg (2016). In the both ways of extrinsic and intrinsic approaches, the method of SPC updates curves on the spherical surfaces to represent the given data and fits curves that satisfy the stationary conditions. For more details, refer to Kim et al. (2020) or Lee et al. (2021a).

The package **spherepc** provides the `SPC()` function for implementing spherical principal curves and the `SPC.Hauberg()` function for principal curves of Hauberg (2016). The usage of the `SPC()` function is as follows.

```
SPC(data, q = 0.05, T = nrow(data), step.size = 1e-3, maxit = 30,
    type = "Intrinsic", thres = 1e-2, deletePoints = FALSE,
    plot.proj = FALSE, kernel = "quartic", col1 = "blue",
    col2 = "green", col3 = "red").
```

The usage of the `SPC.Hauberg()` function is the same as that of the `SPC()` function. Before implementing the `SPC()` and `SPC.Hauberg()` functions, it requires loading the **rgl** (Adler and Murdoch, 2020), **sphereplot** (Robotham, 2013), and **geosphere** (Hijmans et al., 2017) packages. To implement the `SPC()` and `SPC.Hauberg()` functions, we consider the waveform data used in Liu et al. (2017), Kim et al. (2020), and Lee et al. (2021a). The generating equation of waveform is

$$\phi = \alpha \cdot \sin(f\theta \cdot \pi/180) + 10,$$

where ϕ, θ, α, and f denote the longitude, latitude in degrees, amplitude and frequency of the waveform, respectively. θ is uniformly sampled from the interval $[-180, 180]$ and a Gaussian random noise from $N(0, \sigma^2)$ is added on each ϕ where $\sigma = 2, 10$. The generating waveform data and implementations of the `SPC()` and `SPC.Hauberg()` functions are as follows.

```
#### longitude and latitude are expressed in degrees
#### example: waveform data
> n <- 200
> alpha <- 1/3; freq <- 4                    # amplitude and frequency of wave
> sigma1 <- 2; sigma2 <- 10                  # noise levels
> lon <- seq(-180, 180, length.out = n)      # uniformly sampled longitude
> lat <- alpha * 180/pi * sin(freq * lon * pi/180) + 10      # latitude vector
## add Gaussian noises on the latitude vector
> lat1 <- lat + sigma1 * rnorm(length(lon))
> lat2 <- lat + sigma2 * rnorm(length(lon))
> wave1 <- cbind(lon, lat1); wave2 <- cbind(lon, lat2)
## implement Hauberg's principal curves to the waveform data
> SPC.Hauberg(wave1, q = 0.05)
## implement SPC to the (noisy) waveform data
> SPC(wave1, q = 0.05)
> SPC(wave2, q = 0.05)
```

The above codes generate the results in Figure 3. As one can see, the `SPC()` and `SPC.Hauberg()` functions identify the waveform pattern of the simulated data. Especially, the `SPC()` generates a smoother curve. The detailed arguments and outputs of the `SPC()` are described in Tables 5 and 6, respectively, which are the same for the `SPC.Hauberg()`.

Options for spherical principal curves

There are some options for the `SPC()` and `SPC.Hauberg()` functions. In particular, we implement using the arguments `plot.proj` and `deletePoints`, described in Table 5. If `plot.proj = TRUE` is used, then the projection line for each data point is plotted. If the argument `deletePoints = TRUE` is performed, the `SPC()` function deletes the points in curves that do not have adjacent data for each expectation step required to fit the principal curves, returning an open curve, *i.e.*, a curve with endpoints. As a result, the principal curves are more parsimonious since a redundant part of the resulting curves is removed.

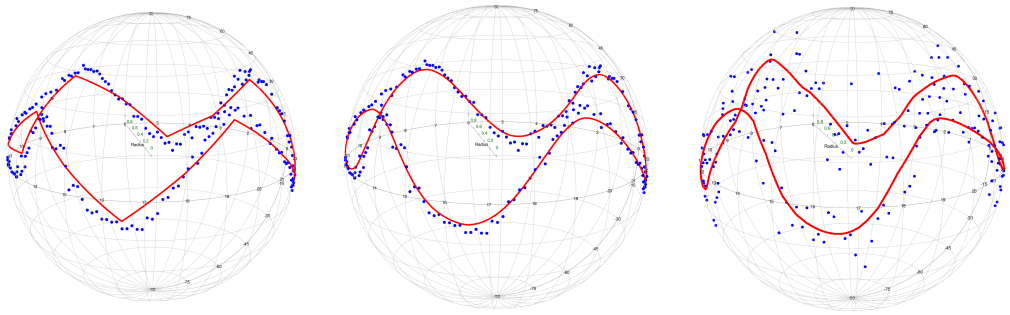


Figure 3: Left and middle: The waveform data (blue) and the results (red) of Hauberg's principal curves (left) and spherical principal curves. Right: The noisy waveform data (blue) and the result (red) of spherical principal curves. All cases are implemented with $q = 0.05$. The two methods find the true waveform of the data well. In particular, the spherical principal curve tends to be smoother.

Argument	Description
`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude (denoted by degrees).
`q`	numeric value of the smoothing parameter. The parameter plays the same role, as the bandwidth does in kernel regression, in the `SPC` function. The value should be a numeric value between 0.01 and 0.5. The default is 0.1.
`T`	the number of points making up the resulting curve. The default is 1000.
`step.size`	step size of the `PrincipalCircle` function. The default is 0.001. The resulting principal circle is used as an initialization of the `SPC` function.
`maxit`	maximum number of iterations. The default is 30.
`type`	type of mean on the sphere. The default is "Intrinsic" and the other choice is "Extrinsic".
`thres`	threshold of the stopping condition. The default is 0.01.
`deletePoints`	logical value. The argument is an option of whether to delete points or not. If `deletePoints` is FALSE, this function leaves the points in curves that do not have adjacent data for each expectation step. As a result, the function usually returns a closed curve, *i.e.* a curve without endpoints. If `deletePoints` is TRUE, this function deletes the points in curves that do not have adjacent data for each expectation step. As a result, The `SPC` function usually returns an open curve, *i.e.* a curve with endpoints. The default is FALSE.
`plot.proj`	logical value. If the argument is TRUE, the projection line for each data point is plotted. The default is FALSE.
`kernel`	kind of kernel function. The default is the quartic kernel, and the alternative is indicator or Gaussian.
`col1`	color of data. The default is blue.
`col2`	color of points in principal curves. The default is green.
`col3`	color of resulting principal curves. The default is red.

Table 5: Arguments of the `SPC()`.

The `SPC.Hauberg()` function also contains the same options. For implementing these two arguments, the following codes are performed through real earthquake data.

```
> data(Earthquake)
# collect spatial locations (longitude and latitude denoted by degrees) of data
> earthquake <- cbind(Earthquake$longitude, Earthquake$latitude)

#### example 1: plot the projection lines (option of plot.proj)
> SPC(earthquake, q = 0.1, plot.proj = TRUE)

#### example 2: open principal curves (option of deletePoints)
> SPC(earthquake, q = 0.04, deletePoints = TRUE)
```

Output	Description
`plot`	plotting of the result in 3D graphics.
`prin.curves`	spatial locations (denoted by degrees) of points in the resulting principal curves.
`line`	connecting lines between points in `prin.curves`.
`converged`	whether or not the algorithm converged.
`iteration`	the number of iterations of the algorithm.
`recon.error`	sum of squared distances between the data and their projections.
`num.dist.pt`	the number of distinct projections.

Table 6: Outputs of the `SPC()`.

The results are illustrated in Figure 4. The left panel shows a closed principal curve (red) with projection lines (black) of each data point onto the curve, and the right panel displays an open principal curve due to the option `deletePoints = TRUE`. It is a parsimonious result because the redundant part on the upper right side of the sphere is removed.

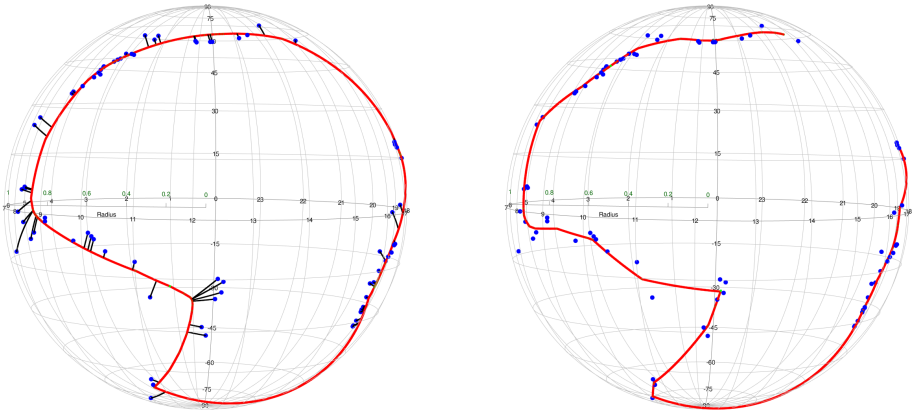


Figure 4: Left: Projection result (black) of SPC with $q = 0.1$. The spherical principal curve (red) continuously represents the earthquake data (blue). Right: The open curve of SPC with $q = 0.04$ and `deletePoints=TRUE`. The less q is, the more the curve overfits the data.

4 Local principal geodesics

Suppose that observations have a non-geodesic structure. Then the PGA may not be beneficial to represent such data because PGA always results in a geodesic line. To overcome this problem, we consider performing PGA locally and repeatedly to detect the non-geodesic and complex structures of data, which can be interpreted as a localized version of the PGA and SPC. The newly proposed method is called local principal geodesics (LPG). The main idea behind the LPG is that non-geodesic structures can be regarded as a part of geodesic when viewed locally. Although there is no reference to the LPG because research on LPG is underway, there is a localized principal curve method on Euclidean space (Einbeck et al., 2005), which is similar to LPG and may share some motivation with the LPG. For more details, refer to Einbeck et al. (2005).

The package **spherepc** offers the `LPG()` function to recognize various data structures, such as spirals, zigzag, and tree data. The usage of the function is

```
LPG(data, scale = 0.04, tau = scale/3, nu = 0, maxpt = 500,
    seed = NULL, kernel = "indicator", thres = 1e-4, col1 = "blue",
    col2 = "green", col3 = "red").
```

Like the previous functions, before the `LPG()` function is implemented, it requires to load the **rgl** (Adler and Murdoch, 2020), **sphereplot** (Robotham, 2013), and **geosphere** (Hijmans et al., 2017) packages. The detailed arguments and outputs of this function are described in Tables 7 and 8. We implement the following code to apply the `LPG()` function to the spiral, zigzag, and tree simulated data illustrated in Figures 5, 6, and 7.

```
## longitude and latitude are expressed in degrees
#### example 1: spiral data
> set.seed(40)
> n <- 900                                    # the number of samples
> sigma1 <- 1; sigma2 <- 2.5;                 # noise levels
> radius <- 73; slope <- pi/16                # radius and slope of the spiral
## polar coordinate of (longitude, latitude)
> r <- runif(n)^(2/3) * radius; theta <- -slope * r + 3
## transform to (longitude, latitude)
> correction <- (0.5 * r/radius + 0.3)        # correction of noise level
> lon1 <- r * cos(theta) + correction * sigma1 * rnorm(n)
> lat1 <- r * sin(theta) + correction * sigma1 * rnorm(n)
> lon2 <- r * cos(theta) + correction * sigma2 * rnorm(n)
> lat2 <- r * sin(theta) + correction * sigma2 * rnorm(n)
> spiral1 <- cbind(lon1, lat1); spiral2 <- cbind(lon2, lat2)
## plot the spiral data
> rgl.sphgrid(col.lat = 'black', col.long = 'black')
> rgl.sphpoints(spiral1, radius = 1, col = 'blue', size = 12)
## implement the LPG to (noisy) spiral data
> LPG(spiral1, scale = 0.06, nu = 0.1, seed = 100)
> LPG(spiral2, scale = 0.12, nu = 0.1, seed = 100)
```

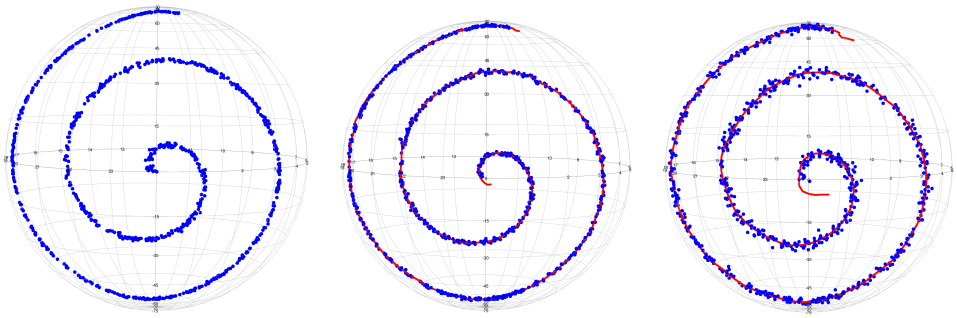


Figure 5: Left: Spiral data (blue) and the result (red) of LPG with scale $= 0.06$ and nu $= 0.1$. Right: Noisy spiral data (blue) and the result (red) of LPG with scale $= 0.12$ and nu $= 0.1$. Local principal geodesics represent the spiral patterns of the (noisy) spiral data. The larger the noise is, the larger scale is required.

```
#### example 2: zigzag data
> set.seed(10)
> n <- 50                                     # the number of samples is 6 * n = 300
> sigma1 <- 2; sigma2 <- 5                    # noise levels
> x1 <- x2 <- x3 <- x4 <- x5 <- x6 <- runif(n) * 20 - 20
> y1 <- x1 + 20 + sigma1 * rnorm(n); y2 <- -x2 + 20 + sigma1 * rnorm(n)
> y3 <- x3 + 60 + sigma1 * rnorm(n); y4 <- -x4 - 20 + sigma1 * rnorm(n)
> y5 <- x5 - 20 + sigma1 * rnorm(n); y6 <- -x6 - 60 + sigma1 * rnorm(n)
> x <- c(x1, x2, x3, x4, x5, x6); y <- c(y1, y2, y3, y4, y5, y6)
> simul.zigzag1 <- cbind(x, y)
## plot the zigzag data
> sphereplot::rgl.sphgrid(col.lat = 'black', col.long = 'black')
> sphereplot::rgl.sphpoints(simul.zigzag1, radius = 1, col = 'blue', size = 12)
## implement the LPG to the zigzag data
> LPG(simul.zigzag1, scale = 0.1, nu = 0.1, maxpt = 45, seed = 50)

## noisy zigzag data
> set.seed(10)
> z1 <- z2 <- z3 <- z4 <- z5 <- z6 <- runif(n) * 20 - 20
> w1 <- z1 + 20 + sigma2 * rnorm(n); w2 <- -z2 + 20 + sigma2 * rnorm(n)
> w3 <- z3 + 60 + sigma2 * rnorm(n); w4 <- -z4 - 20 + sigma2 * rnorm(n)
> w5 <- z5 - 20 + sigma2 * rnorm(n); w6 <- -z6 - 60 + sigma2 * rnorm(n)
> z <- c(z1, z2, z3, z4, z5, z6); w <- c(w1, w2, w3, w4, w5, w6)
> simul.zigzag2 <- cbind(z, w)
## implement the LPG to the noisy zigzag data
```

```
> LPG(simul.zigzag2, scale = 0.2, nu = 0.1, maxpt = 18, seed = 20)
```

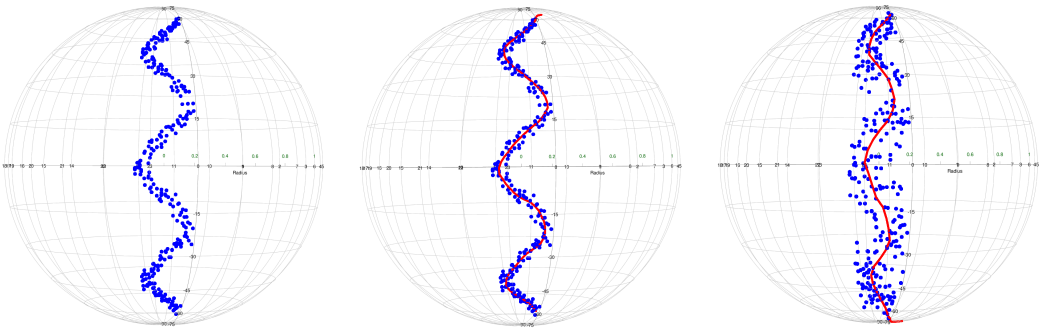


Figure 6: Left: zigzag data (blue). Middle: zigzag data (blue) and the result (red) of with scale $= 0.1$ and nu $= 0.1$. Right: Noisy zigzag data (blue) and the result (red) of LPG with scale $= 0.2$, and nu $= 0.1$. Local principal geodesics extract the zigzag structures of the (noisy) zigzag data properly. The larger the noise is, the larger scale is needed.

Note that the `LPG()` function may return several curves. We now implement the function in a complex simulation dataset composed of several curves. As shown in the left panel of Figure 7, the tree object has twenty-six geodesic (linear) structures consisting of one stem, five branches, and twenty subbranches. It is not informative to show the generating formula for the tree dataset. Instead, we provide its generating code with explanatory notes as follows.

```
#### example 3: tree dataset
## the tree dataset consists of stem, branches and subbranches
## generate stem
> set.seed(10)
> n1 <- 200; n2 <- 100; n3 <- 15     # the number of samples in
                                     # a stem, a branch, and a subbrach
> sigma1 <- 0.1; sigma2 <- 0.05; sigma3 <- 0.01          # noise levels
> noise1 <- sigma1 * rnorm(n1); noise2 <- sigma2 * rnorm(n2)
> noise3 <- sigma3 * rnorm(n3)
> l1 <- 70; l2 <- 20; l3 <- 1        # length of stem, branches, and subbranches
> rep1 <- l1 * runif(n1)             # repeated part of stem
> stem <- cbind(0 + noise1, rep1 - 10)
## generate branch
> rep2 <- l2 * runif(n2)             # repeated part of branch
> branch1 <- cbind(-rep2, rep2 + 10 + noise2); branch2 <- cbind(rep2, rep2 + noise2)
> branch3 <- cbind(rep2, rep2 + 20 + noise2)
> branch4 <- cbind(rep2, rep2 + 40 + noise2)
> branch5 <- cbind(-rep2, rep2 + 30 + noise2)
> branch <- rbind(branch1, branch2, branch3, branch4, branch5)
## generate subbranches
> rep3 <- l3 * runif(n3)             # repeated part in subbranches
> branches1 <- cbind(rep3 - 10, rep3 + 20 + noise3)
> branches2 <- cbind(-rep3 + 10, rep3 + 10 + noise3)
> branches3 <- cbind(rep3 - 14, rep3 + 24 + noise3)
> branches4 <- cbind(-rep3 + 14, rep3 + 14 + noise3)
> branches5 <- cbind(-rep3 - 12, -rep3 + 22 + noise3)
> branches6 <- cbind(rep3 + 12, -rep3 + 12 + noise3)
> branches7 <- cbind(-rep3 - 16, -rep3 + 26 + noise3)
> branches8 <- cbind(rep3 + 16, -rep3 + 16 + noise3)
> branches9 <- cbind(rep3 + 10, -rep3 + 50 + noise3)
> branches10 <- cbind(-rep3 - 10, -rep3 + 40 + noise3)
> branches11 <- cbind(-rep3 + 12, rep3 + 52 + noise3)
> branches12 <- cbind(rep3 - 12, rep3 + 42 + noise3)
> branches13 <- cbind(rep3 + 14, -rep3 + 54 + noise3)
> branches14 <- cbind(-rep3 - 14, -rep3 + 44 + noise3)
> branches15 <- cbind(-rep3 + 16, rep3 + 56 + noise3)
> branches16 <- cbind(rep3 - 16, rep3 + 46 + noise3)
> branches17 <- cbind(-rep3 + 10, rep3 + 30 + noise3)
```

```
> branches18 <- cbind(-rep3 + 14, rep3 + 34 + noise3)
> branches19 <- cbind(rep3 + 16, -rep3 + 36 + noise3)
> branches20 <- cbind(rep3 + 12, -rep3 + 32 + noise3)
> sub.branches <- rbind(branches1, branches2, branches3, branches4, branches5,
+ branches6, branches7, branches8, branches9, branches10, branches11, branches12,
+ branches13, branches14, branches15, branches16, branches17, branches18,
+ branches19, branches20)
## tree dataset consists of stem, branch, and subbranches
> tree <- rbind(stem, branch, sub.branches)
## plot the tree dataset
> sphereplot::rgl.sphgrid(col.lat = 'black', col.long = 'black')
> sphereplot::rgl.sphpoints(tree, radius = 1, col = 'blue', size = 12)
## implement the LPG function to the tree dataset
> LPG(tree, scale = 0.03, nu = 0.2, seed = 10)
```

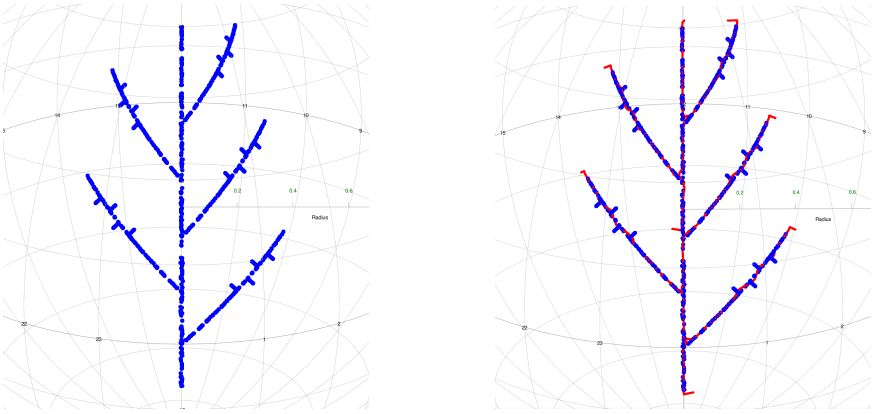


Figure 7: Tree data (blue) and the result (red) of LPG with scale $= 0.03$ and nu=0.2. The LPG function captures the complex structures of the data well, provided that scale and nu are properly chosen.

As displayed in Figures 5, 6, and 7, the `LPG()` function identifies the non-geodesic or complex patterns of the simulated datasets well as long as the parameters of scale and nu are properly chosen. The arguments and outputs of the function are respectively described in Tables 7 and 8.

Argument	Description
`data`	matrix or data frame consisting of spatial locations with two columns. Each row represents longitude and latitude (denoted by degrees).
`scale`	scale parameter for this function. The argument is the degree to which the `LPG` function expresses data locally; thus, as the `scale` grows, the result of the `LPG` becomes similar to that of the `PGA` function. The default is 0.4.
`tau`	forwarding or backwarding distance of each step. It is empirically recommended to choose a third of `scale`, which is the default of this argument.
`nu`	parameter to alleviate bias of resulting curves. `nu` represents the viscosity of the given data and it should be selected in $[0, 1)$. The default is zero. When `nu` is close to 1, the curve usually swirls similarly to the motion of a large viscous fluid. The argument `maxpt` can control the swirling.
`maxpt`	maximum number of points in each curve. The default is 500.
`seed`	random seed number.
`kernel`	kind of kernel function. The default is the indicator kernel, and the alternative is quartic or Gaussian.
`thres`	threshold of the stopping condition for the `IntrinsicMean` function in the process of the `LPG` function. The default is 1e-4.
`col1`	color of data. The default is blue.
`col2`	color of points in the resulting principal curves. The default is green.
`col3`	color of the resulting curves. The default is red.

Table 7: Arguments of the `LPG()`.

Output	Description
`plot`	plotting of the result in 3D graphics.
`num.curves`	the number of resulting curves.
`prin.curves`	spatial locations (represented by degrees) of points in the resulting curves.
`line`	connecting lines between points in `prin.curves`.

Table 8: Outputs of the `LPG()`.

5 Application

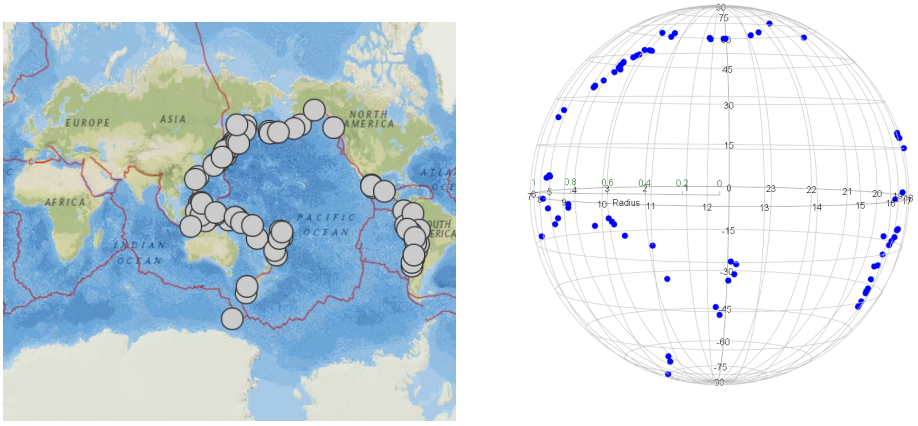


Figure 8: The distribution of significant earthquakes (8+ Mb magnitude), and their three-dimensional visualization.

We use earthquake data from the U.S. Geological Survey, which has collected significant earthquakes (8+ Mb magnitude) around the Pacific Ocean since 1900. As shown in Figure 8, the data contain 77 observations distributed in the borders between the Eurasian, Pacific, North American, and Nazca tectonic plates. The data have three features: the observations are distributed globally, scattered, and form non-geodesic structures. Because the tectonic plates are constantly moving in different directions, identifying the hidden patterns of borders is useful in geostatistics and seismology, as noted in Biau and Fischer (2011); Mardia (2014). It can be possible to identify the borders of plates by applying dimension reduction methods to the earthquake data.

To apply the aforementioned dimension reduction methods to the earthquake data, we use the following code.

```
> data(Earthquake)
#### collect spatial locations (longitude and latitude by degrees) of data
> earthquake <- cbind(Earthquake$longitude, Earthquake$latitude)

#### example 1: principal geodesic analysis (PGA)
> PGA(earthquake)

#### example 2: principal circle
## get center and radius of principal circle
> circle <- PrincipalCircle(earthquake)
## generate the principal circle
> PC <- GenerateCircle(circle[1:2], circle[3], T = 1000)
## plot the principal circle
> sphereplot::rgl.sphgrid(col.long = "black", col.lat = "black")
> sphereplot::rgl.sphpoints(earthquake, radius = 1, col = "blue", size = 12)
> sphereplot::rgl.sphpoints(PC, radius = 1, col = "red", size = 9)
```

Examples 1 and 2 implement the principal geodesic and the principal circle, respectively. As illustrated in Figure 9, the principal geodesic (left) fails to identify the variations of the earthquake data. The principal circle (right) captures the global trend of the data, whereas the circle could not extract the local variations of the data.

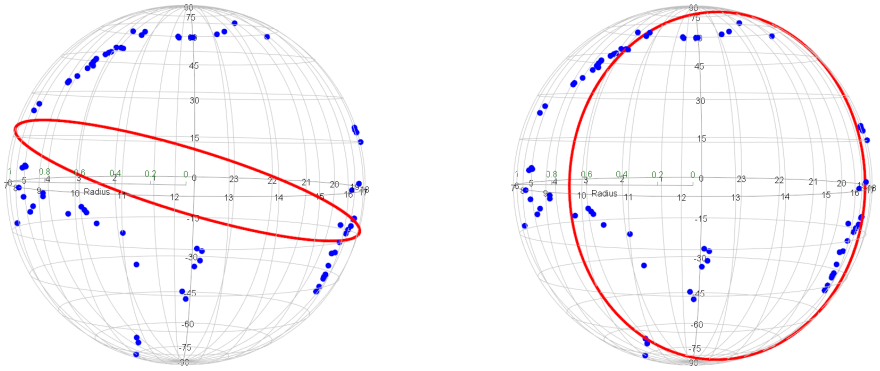


Figure 9: Earthquake data (blue) and the results (red) of the principal geodesic analysis and principal circle, from left to right. The principal geodesic fails to find the non-geodesic feature of the data, and the principal circle captures the circular pattern but cannot identify the local variations of the data.

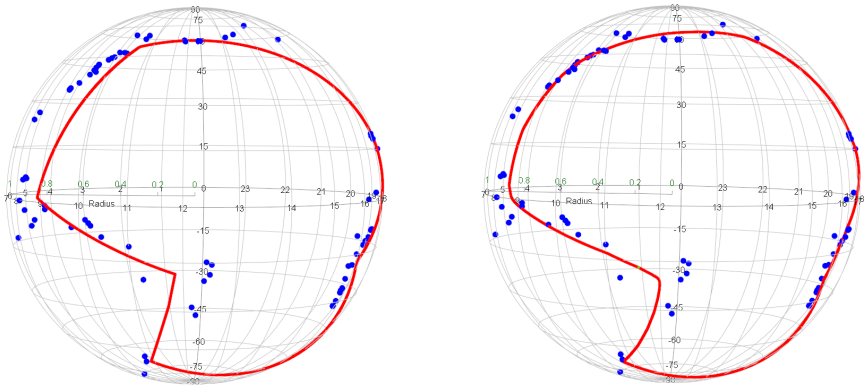


Figure 10: Earthquake data (blue) and implementation results (red) with $q = 0.1$ of the SPC.Hauberg and SPC functions, respectively, from left to right. Both methods can represent the non-geodesic feature of the earthquake data. The spherical principal curve particularly tends to be smoother.

```
#### example 3: spherical principal curves and principal curves of Hauberg
> SPC.Hauberg(earthquake, q = 0.1) # principal curves of Hauberg
> SPC(earthquake, q = 0.1)         # spherical principal curves
```

Example 3 fits the spherical principal curve and Hauberg's principal curve with $q = 0.1$. As shown in Figure 10, both methods identify the curved feature of the earthquake data. The spherical principal curve particularly tends to be more continuous than Hauberg's principal curve.

```
#### example 4: spherical principal curves with q = 0.15, 0.1, 0.03, and 0.02
> SPC(earthquake, q = 0.15)
> SPC(earthquake, q = 0.1)
> SPC(earthquake, q = 0.03)
> SPC(earthquake, q = 0.02)
```

Example 4 applies the spherical principal curve to the earthquake data with varying $q = 0.15, 0.1, 0.03, 0.02$. The parameter q plays a role in the bandwidth of the `SPC()` function. As shown in Figure 11, the smaller q is, the rougher the curve is. On the contrary, the larger q is, the smoother the curve is.

```
#### example 5: local principal geodesics (LPG)
> LPG(earthquake, scale = 0.5, nu = 0.2, maxpt = 20, seed = 50)
> LPG(earthquake, scale = 0.4, nu = 0.3, maxpt = 22, seed = 50)
```

Lastly, example 5 implements the `LPG()` function with different `scale` and `nu`. As shown in Figure 12, the function represents the curved pattern of the data, illustrating the slightly different features.

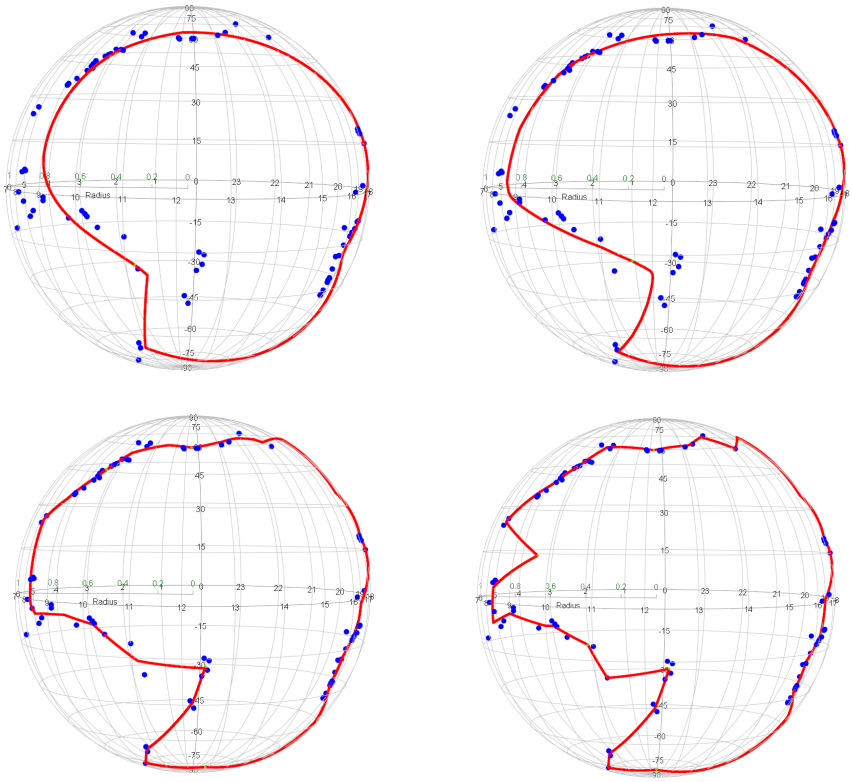


Figure 11: From left to right and top to bottom: Earthquake data (blue) and the results (red) of the SPC with $q = 0.15,\ 0.1,\ 0.03$ and 0.02. The larger the parameter q is, the smoother the curve is, while it tends to underfit the data. Conversely, the smaller the parameter q is, the rougher the curve is.

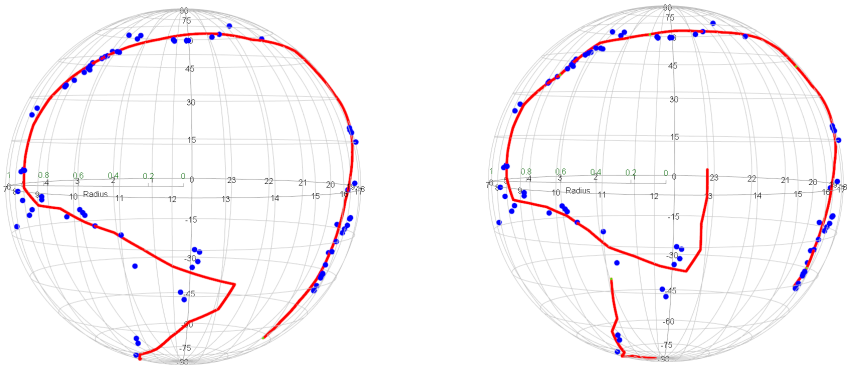


Figure 12: From left to right, earthquake data (blue) and the results of the LPG function with scale $= 0.5$, nu=0.2 and scale $= 0.4$, nu=0.3. Both the local principal geodesics implemented by different parameters recognize the non-geodesic and scattered pattern of the data, illustrating the different features.

6 Conclusions

In this paper, the R package **spherepc** has implemented various dimension reduction methods on a sphere. It includes not only principal geodesic analysis (PGA), principal circle, and principal curves of Hauberg (2016) as existing methods but also spherical principal curves (SPC) and local principal geodesics (LPG) as new approaches. The **spherepc** package has demonstrated its usefulness by applying the functions to several simulation examples and real earthquake data. We believe that the **spherepc** is helpful for applications in various fields, ranging from statistics to engineering, such as geostatistics, image analysis, pattern recognition, and machine learning.

7 Acknowledgments

This research was supported by the National Research Foundation of Korea (NRF) funded by the Korea government (2020R1A4A1018207; 2021R1A2C1091357).

Bibliography

D. Adler and D. Murdoch. ***rgl**: 3D Visualization Using OpenGL*, 2020. URL https://cran.r-project.org/package=rgl. R package version 0.100.50. [p168, 169, 171, 173]

P. Benner, V. Mehrmann, and D. C. Sorensen. *Dimension Reduction of Large-scale Systems*, volume 45. Springer, 2005. [p167]

G. Biau and A. Fischer. Parameter selection for principal curves. *IEEE Transactions on Information Theory*, 58(3):1924–1939, 2011. [p177]

J. Einbeck, G. Tutz, and L. Evers. Local principal curves. *Statistics and Computing*, 15(4):301–313, 2005. [p173]

J. Einbeck, L. Evers, and M. J. Einbeck. ***LPCM**: Local Principal Curve Method*, 2015. URL https://cran.r-project.org/package=LPCM. R package version 0.46-7. [p167]

P. T. Fletcher, C. Lu, S. M. Pizer, and S. Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, 23(8):995–1005, 2004. [p167]

T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406): 502–516, 1989. [p167, 170]

T. Hastie and A. Weingessel. ***princurve**: Fits a Principal Curve in Arbitrary Dimension*, 2015. URL https://cran.r-project.org/package=princurve. R package version 2.16. [p167]

S. Hauberg. Principal curves on riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1915, 2016. [p167, 171, 179]

R. J. Hijmans, E. Williams, and C. Vennes. ***geosphere**: Spherical Trigonometry*, 2017. URL https://cran.r-project.org/package=geosphere. R package version 1.5-10. [p168, 171, 173]

S. Huckemann, T. Hotz, and A. Munk. Intrinsic shape analysis: Geodesic pca for riemannian manifolds modulo isometric lie group actions. *Statistica Sinica*, pages 1–58, 2010. [p167]

J.-H. Kim, J. Lee, and H.-S. Oh. Spherical principal curves. *arXiv preprint arXiv:2003.02578*, 2020. [p169, 171]

J. Lee, J.-H. Kim, and H.-S. Oh. Spherical principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(6):2165–2171, 2021a. [p167, 169, 171]

J. Lee, J.-H. Kim, and H.-S. Oh. Supplementary material for spherical principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021b. Available online. [p169]

H. Liu, Z. Yao, S. Leung, and T. F. Chan. A level set based variational principal flow method for nonparametric dimension reduction on riemannian manifolds. *SIAM Journal on Scientific Computing*, 39(4):A1616–A1646, 2017. [p167, 171]

K. V. Mardia. *Statistics of Directional Data*. Academic press, 2014. [p177]

V. M. Panaretos, T. Pham, and Z. Yao. Principal flows. *Journal of the American Statistical Association*, 109 (505):424–436, 2014. [p167]

A. Robotham. ***sphereplot**: Spherical Plotting*, 2013. URL https://cran.r-project.org/package=sphereplot. R package version 1.5. [p168, 169, 171, 173]

Jongmin Lee
Department of Statistics
Seoul National University
Seoul 08826, Korea
(ORCiD 0000-0003-1723-4615)
jongminlee9218@gmail.com

Jang-Hyun Kim
Department of Computer Science and Engineering
Seoul National University
Seoul 08826, Korea
(ORCiD 0000-0001-8433-2712)
kimjanghyun1230@gmail.com

Hee-Seok Oh
Department of Statistics
Seoul National University
Seoul 08826, Korea
(ORCiD 0000-0002-1501-0530)
heeseok@stats.snu.ac.kr

The smoots Package in R for Semiparametric Modeling of Trend Stationary Time Series

by Yuanhua Feng, Thomas Gries, Sebastian Letmathe and Dominik Schulz

Abstract This paper is an introduction to the new package in R called **smoots** (smoothing time series), developed for data-driven local polynomial smoothing of trend-stationary time series. Functions for data-driven estimation of the first and second derivatives of the trend are also built-in. It is first applied to monthly changes of the global temperature. The quarterly US-GDP series shows that this package can also be well applied to a semiparametric multiplicative component model for non-negative time series via the log-transformation. Furthermore, we introduced a semiparametric Log-GARCH and a semiparametric Log-ACD model, which can be easily estimated by the **smoots** package. Of course, this package applies to suitable time series from any other research area. The **smoots** package also provides a useful tool for teaching time series analysis, because many practical time series follow an additive or a multiplicative component model.

1 Introduction

This paper provides an introduction to a new package in R called **smoots** (version 1.0.1, Feng and Schulz, 2019) for data-driven local polynomial smoothing of trend-stationary time series. This package is developed based on the R codes for the practical implementation of the proposals in Feng et al. (2020). Detailed discussion on the methodology and the development of the algorithms may be found in that work. Here, the main algorithm is a fully data-driven IPI (iterative plug-in, Gasser et al., 1991) approach for estimating the trend under stationary time series errors, where the variance factor in the asymptotically optimal bandwidth is estimated by another IPI-approach for a lag-window estimator of the spectral density following Bühlmann (1996). Numerous sub-algorithms determined by different options, such as the choice of the order of local polynomial, the choice of the kernel weighting functions and the choice of the so-called inflation factors for estimating the bias in the asymptotically optimal bandwidth, are included. Further functions for data-driven estimation of the first and second derivatives of the trend are also developed by adapting the idea of Feng (2007). A related proposal may be found in Francisco-Fernández et al. (2004). The algorithms included in the **smoots** package differ to their proposal in several ways. (1) The IPI-algorithm is usually superior to the DPI (see Beran et al., 2009, for detailed discussion in models with i.i.d. errors). (2) In Francisco-Fernández et al. (2004) the variance factor is estimated under an AR(1) model, which is usually a misspecification. (3) Data-driven estimation of the derivatives are also included in the current package. Moreover, the user can obtain any estimate with fixed bandwidths chosen beforehand. A function for kernel smoothing is also built-in for comparison.

The **smoots** package complements the Comprehensive R Archive Network (CRAN), as already existing base R functions or CRAN packages for local polynomial smoothing either do not implement an automated bandwidth selection or, if such bandwidth algorithms do exist, they are only suitable for data with i.i.d. (independently and identically distributed) errors, which is an assumption that is often violated for time series. In more detail, notwithstanding that the **stats** package offers the functions `lowess()` and `loess()` for computing the local polynomial estimates of the regression function given one or multiple predictor variables, a smoothing bandwidth has to be selected arbitrarily in both cases. Similar functionality is provided by the **locpol** (Ojeda Cabrera, 2018), **KernSmooth** (Wand, 2021) and **lokern** (Herrmann and Maechler, 2021) packages, however, derivative estimation approaches and various bandwidth selection algorithms for the regression function, such as the leave-one-out estimator (Allen, 1974) and the plug-in algorithm by Ruppert et al. (1995), are built into them. Moreover, within **lokern**, the bandwidth for estimating the first or second derivative of the regression function can be selected automatically. Nevertheless, these traditional data-driven bandwidth selection approaches are only suitable in case of data with i.i.d. errors (Hart, 1991; Altman, 1990; Opsomer, 1997) and can therefore often not be considered for time series. Explicit CRAN packages for detrending a time series nonparametrically are **rmaf** (Qiu, 2015) and **mFilter** (Balcilar, 2019). While with **rmaf**, a fully data-driven refined moving average and a cubic smoothing spline, which requires a manually selected smoothness parameter, can be applied, the trend as well as the cyclical component of a time series can be extracted with the filters implemented in the **mFilter** package like the Baxter-King (Baxter and King, 1999) or Hodrick-Prescott (Hodrick and Prescott, 1997) filters among others. However, all filtering functions in the **mFilter** package are not fully data-driven. Instead, they make use of default values

recommended within scientific literature. Thus, to our knowledge, there are not any packages on CRAN that implement a data-driven local polynomial regression for the trend of a time series or its derivatives.

The methodological and computational background for the **smoots** package is summarized briefly hereinafter. For further details we refer the reader to Feng et al. (2020) and references therein. Our focus is to illustrate the wide application spectrum of the **smoots** package for semiparametric modeling of time series. We propose to estimate the nonparametric trend using the current package in the first stage and to fit e.g. an ARMA model to the residuals with the standard `arima()` function of the **stats** package in R. This idea is first shown with monthly Northern Hemisphere temperature changes data obtained from the website of the National Aeronautics and Space Administration (NASA). Then the proposal is applied to the log-transformation of the quarterly US-GDP data, retrieved from the website of the Federal Reserve Bank of St. Louis, using a semiparametric log-local-linear growth model. Moreover, two new semiparametric models for financial time series are proposed to show further application of this package. Firstly, a Semi-Log-GARCH model is defined by introducing a scale function into the Log-GARCH (logarithmic GARCH) (Pantula, 1986; Geweke, 1986; Milhøj, 1987). The latter is a special extension of the seminal ARCH (autoregressive conditional heteroskedasticity, Engle, 1982) and GARCH (generalized ARCH, Bollerslev, 1986) models and is just a slightly restricted ARMA model for the log-transformation of the squared returns (see e.g. Sucarrat, 2019). The usefulness of the Log-GARCH has recently been rediscovered by Francq et al. (2013). The application of this proposal is illustrated by the DAX returns collected from Yahoo Finance. Secondly, a Semi-Log-ACD model is proposed as an extension of the Type I Log-ACD (Bauwens and Giot, 2000), which is closely related to the Semi-Log-GARCH. Like the ACD (autoregressive conditional duration, Engle and Russell, 1998) model, the Semi-Log-ACD can be applied to different non-negative financial data, such as realized volatilities and trading volumes. In this paper, this new model is applied to the CBOE Volatility Index (VIX) obtained from Yahoo Finance. Datasets for all of those examples are built in the proposed package. The **smoots** package can be applied to suitable time series from other research areas. In addition, the **smoots** package provides a useful tool for teaching time series analysis, which helps a lecturer to obtain automatically detrended real data examples to show the application of parametric time series models.

The methods and IPI-algorithms are summarized in the following two sections. The general usage of the R functions is then described in another section. Three further sections illustrate the applications of those functions in simple cases and with respect to the Semi-Log-GARCH and the Semi-Log-ACD.

2 Local polynomial regression for time series

The main algorithm of the **smoots** package is a fully automatic non-parametric procedure for estimating a deterministic trend in an additive time series model with observations y_t, $t = 1, ..., n$. The data under consideration can for example be from environmental statistics, economics or finance. The basic nonparametric time series model is defined by

$$y_t = m(x_t) + \xi_t, \tag{1}$$

where $x_t = t/n$ denotes the rescaled time, m is a smooth function and ξ_t is a zero mean stationary process. This model defines a nonparametric approach for trend-stationary time series. Let $\gamma_\xi(\tau)$ denote the acf (autocovariances) of ξ_t. Under the regularity conditions given in Bühlmann (1996), a data-driven IPI-algorithm for estimating the variance factor in the asymptotically optimal bandwidth can be developed. As indicated by Feng et al. (2020), the required conditions are fulfilled by an ARMA process with finite eighth moment. However, models with long-memory errors are excluded by those assumptions.

The local polynomial estimator of $m^{(\nu)}(x)$, the ν-th derivative of m, is obtained by minimizing

$$Q = \sum_{t=1}^{n} \left\{ y_t - \sum_{j=0}^{p} b_j(x) (x_t - x)^j \right\}^2 W\left(\frac{x_t - x}{h} \right), \tag{2}$$

where h is the bandwidth, W is a second order kernel function with compact support $[-1, 1]$ and is used here as the weighting function, and p is the order of polynomial. It is assumed that $p - \nu$ is odd. We have $\hat{m}^{(\nu)}(x) = \nu! \hat{b}_\nu(x)$, which is asymptotically equivalent to some kernel regression with a k-th order kernel $K(u)$ and automatic boundary correction, where $k = p + 1$. In this paper, the following weighting functions are considered:

$$W(u) = C_\mu \left(1 - u^2\right)^\mu \mathbb{I}_{[-1,1]}(u), \tag{3}$$

where C_μ is a standardization constant that is indeed irrelevant for calculating the weights in (3) and $\mu = 0, 1, ...$, is a smoothness parameter. For automatic bandwidth selection using the **smoots** package, only $\mu = 0, 1, 2$ and 3 are allowed, corresponding to the use of the uniform, Epanechnikov, bisquare and triweight kernels, respectively.

An IPI-algorithm is developed based on the asymptotically optimal bandwidth h_A, which is the minimizer of the AMISE (asymptotic mean integrated squared error):

$$\text{AMISE}(h) = h^{2(k-\nu)} \frac{I\left[m^{(k)}\right]\beta^2}{(k!)^2} + \frac{2\pi c_f (d_b - c_b) R(K)}{nh^{2\nu+1}}, \tag{4}$$

where $I\left[m^{(k)}\right] = \int_{c_b}^{d_b} \left[m^{(k)}(x)\right]^2 dx$, $\beta = \int_{-1}^{1} u^k K(u)\, du$, $R(K) = \int_{-1}^{1} K^2(u)\, du$, $c_f = f(0)$ is the value of the spectral density at the frequency zero and $0 \le c_b < d_b \le 1$ are introduced to reduce the effect of the estimates at the boundary points. Then we have

$$h_A = n^{-\frac{1}{2k+1}} \left(\frac{2\nu+1}{2(k-\nu)} \frac{2\pi c_f (k!)^2 (d_b - c_b) R(K)}{I\left[m^{(k)}\right]\beta^2} \right)^{\frac{1}{2k+1}}. \tag{5}$$

After estimating and removing the nonparametric trend, any suitable parametric model can be fitted to the residuals for further econometric analysis. For instance, we can assume that ξ_t follows an ARMA model:

$$\xi_t = \varphi_1 \xi_{t-1} + ... + \varphi_r \xi_{t-r} + \psi_1 \varepsilon_{t-1} + ... + \psi_s \varepsilon_{t-s} + \varepsilon_t, \tag{6}$$

where ε_t are i.i.d. innovations. A semiparametric ARMA (Semi-ARMA) model is then defined by (1) and (6).

3 The proposed IPI-algorithms

Since both c_f and $I\left[m^{(k)}\right]$ in (5) are unknown, we need to obtain and insert appropriate estimates of these two quantities into this formula to achieve a selected bandwidth $\hat{h}_A$. However, the estimation of c_f and $I\left[m^{(k)}\right]$ requires the use of two additional bandwidths. Hence, iterative bandwidth selection procedures should be employed. The estimation of $I\left[m^{(k)}\right]$ is not influenced by the correlation structure, which can be simply obtained by means of established IPI-algorithms for models with independent errors. Consequently, solely a comprehensive review of the estimation approach for c_f will be given.

Data-driven estimation of c_f

Let $r_{t,\mathrm{V}} = y_t - \hat{m}_{\mathrm{V}}$ denote the residuals obtained from a pilot estimate using a bandwidth h_{V} and denote the sample acf calculated from $r_{t,\mathrm{V}}$ by $\hat{\gamma}(l)$. In this paper we propose to use the following lag-window estimator of c_f:

$$\hat{c}_{f,\mathrm{M}} = \frac{1}{2\pi} \sum_{l=-M}^{M} w_l \hat{\gamma}(l), \tag{7}$$

where $w_l = l/(M + 0.5)$ are weights calculated according to some lag-window with the window-width M. And, M will be selected by the following IPI-algorithm proposed by Feng et al. (2020), which is adjusted from that of Bühlmann (1996).

i) Let $M_0 = [n/2]$ be the initial window-width, where $[\cdot]$ denotes the integer part.

ii) Global steps: Estimate $\int \left(f(\lambda)^2\right) d\lambda$ in the j-th iteration following Bühlmann (1996). Denote by $\int f^{(1)}(\lambda)\, d\lambda$ the integral of the first generalized derivative of $f(\lambda)$. Estimate it using the window-width $M_j' = \left[M_{j-1}/n^{2/21}\right]$, the proposal in Bühlmann (1996) and the Bartlett-window. Obtain M_j by inserting this quantity into Eq. (5) in Bühlmann (1996). Carry out this procedure iteratively until convergence is reached or until a maximum of 20 iterations. Denote the selected M by $\hat{M}_G$.

iii) Local adaptation at $\lambda = 0$: Calculate $\int f^{(1)}(\lambda)\, d\lambda$ again using $M' = \left[\hat{M}_G/n^{2/21}\right]$. Obtain the finally selected window-width $\hat{M}$ by inserting the estimates into the formula of the local optimal bandwidth at $\lambda = 0$ in (5) of Bühlmann (1996).

It is proposed to use the Bartlett-window throughout the whole algorithm for simplicity. If $\hat{M}$ converges, it is usually not affected by the starting value M_0. Hence, any $1 \leq M_0 \leq [n/2]$ can be used in the proposed algorithm.

The IPI-algorithm for estimating m

In this subsection the data-driven IPI-algorithm for selecting the bandwidth for local linear and local cubic estimators $\hat{m}$, with $k = 2$ and 4, respectively, under correlated errors will be described. Note that, although the variance factor c_f can be estimated well from residuals of $\hat{m}$, Feng et al. (2020) showed that the asymptotically optimal bandwidth for estimating c_f should be $\text{CF} \cdot h_A$, not h_A itself, where

$$\text{CF} = \left\{ 2k \left[\frac{2K(0)}{R(K)} - 1 \right] \right\}^{1/(2k+1)} \tag{8}$$

is a correction factor to obtain the asymptotically optimal bandwidth for estimating $\hat{c}_f$ from $\hat{h}$, which is the same as given in Feng and Heiler (2009) and is always bigger than 1. Theoretically, c_f should be estimated from $r_{t,\text{V}}$ obtained with the bandwidth $\text{CF} \cdot \hat{h}$. The proposed main IPI-algorithm for selecting the bandwidth proceeds as follows.

i) Start with an initial bandwidth h_0 given beforehand.

ii) Obtain $r_{t,\text{V}}$ using h_{j-1} or $\text{CF} \cdot h_{j-1}$ and estimate c_f from $r_{t,\text{V}}$ as proposed above.

iii) Let $\alpha = 5/7$ or $5/9$ for $p = 1$, and $\alpha = 9/11$ or $9/13$ for $p = 3$, respectively. Estimate $I\left[m^{(k)}\right]$ with $h_{d,j} = h_{j-1}^{\alpha}$ and a local polynomial of order $p_d = p + 2$. We obtain

$$h_j = \left(\frac{[k!]^2}{2k\beta^2} \frac{2\pi \hat{c}_f (d_b - c_b) R(K)}{I\left[\hat{m}^{(k)}\right]} \right)^{1/(2k+1)} \cdot n^{-1/(2k+1)}. \tag{9}$$

iv) Increase j by 1. Repetitively carry out Steps *ii*) and *iii*) until convergence is reached or until for example $J = 20$ iterations are achieved. Let $\hat{h} = h_j$ be the selected bandwidth.

In the developed package the initial bandwidth $h_0 = 0.15$ for both $p = 1$ and $p = 3$ is used. Also, for both $p = 1$ and $p = 3$, the default value $c_b = 1 - d_b = 0.05$ is proposed to reduce the effect of the estimates at the boundary points. That is, the bandwidth is selected only using 90% of the observations in the middle part. The bandwidth $h_{d,j} = h_{j-1}^{\alpha}$ for estimating the k-th derivative is roughly fixed using a so-called EIM (exponential inflation method). The first α value is chosen so that $I\left[\hat{m}^{(k)}\right]$ and hence $\hat{h}$ will achieve their optimal rates of convergence. Now, the algorithm will be denoted by **AlgA**. If the second α value is used, $\hat{m}^{(k)}$, but not $\hat{h}$, will achieve its optimal rate of convergence. This algorithm is called **AlgB**. And $\hat{h}$ selected by **AlgB** is larger than that by **AlgA**. For further details on those topics we refer the reader to Feng et al. (2020).

Data-driven estimation of m' and m''

The proposed IPI-algorithm can be easily adapted to select bandwidths for estimating $\hat{m}^{(\nu)}$. In the following, only the cases with $\nu = 1$ or 2 are considered, where c_f is estimated by means of a data-driven pilot estimate $\hat{m}$ of the order p_p, say. Then $m^{(\nu)}$ will be estimated with $p = \nu + 1$ and $k = \nu + 2$. As before, $m^{(k)}$ for calculating the bias factor will be estimated with $p_d = p + 2$ and a correspondingly inflated bandwidth. This leads to the following two-stage procedure.

i) In the first stage $\hat{c}_f$ is obtained by the main IPI-algorithm with $p_p = 1$ or $p_p = 3$.

ii) Then an IPI-procedure as proposed above to select the bandwidth for estimating $m^{(\nu)}$ according to (5) is carried out with fixed $\hat{c}_f$ obtained in i).

Note that $\hat{m}^{(\nu)}$ is asymptotically equivalent to some kernel estimator with boundary correction. Explicit forms of the equivalent kernels for estimating $m^{(\nu)}$ in the middle part may be found in Müller (1988). The corresponding inflation factors (i.e. the α values) are determined by p or k.

4 Practical implementation in R

The package **smoots** developed based on the algorithms described in the last section consists of five directly usable functions, two S3 methods and four datasets as examples. The first four functions are called `msmooth()`, `tsmooth()`, `gsmooth()` and `knsmooth()`, designed for estimating the trend in different ways. Data-driven estimation can be carried out by each of the first two functions, where `msmooth()` is a user-friendlier simplified version of `tsmooth()`. Local polynomial estimation of $m^{(\nu)}$ and kernel smoothing of m with an arbitrary bandwidth fixed beforehand can be carried out by `gsmooth()` and `knsmooth()`, respectively. In the first case, one should choose p such that $p - \nu$ is odd to avoid the boundary problem. Those functions allow a flexible application of this package by an experienced user. Data-driven estimation of the first or second derivative can be obtained by `dsmooth()`.

The functions for selecting the bandwidth will be described in more detail. Moreover, for simplicity, shared arguments between functions will only be discussed once. With

```
tsmooth(y, p, mu, Mcf, InfR, bStart, bvc, bb, cb, method),
```

the trend in equidistant and trend-stationary time series with short-memory can be estimated via an automated local polynomial. Its arguments are as follows.

- `y` is the input time series.
- `p` reflects the order of polynomial and currently only `p = 1` (local linear) and `p = 3` (local cubic) are selectable.
- `mu` corresponds to the smoothness parameter μ of the second order kernel function (3) used for weighting. Only `0`, `1`, `2` and `3` are valid options.
- `Mcf` defines the method for estimating c_f. For `Mcf = "NP"`, the default, c_f is estimated nonparametrically as described in the section *Data-driven estimation of c_f*. If `"AR"`, `"MA"` or `"ARMA"` are selected, ξ_t in model (1) is assumed to follow an AR, MA or ARMA process, respectively, during the bandwidth selection and thus, c_f is estimated parametrically.
- `InfR` sets the inflation rate α considered for the bandwidth (see also Step iii) in the section *The IPI-algorithm for estimating m*). The options are `InfR = "Opt"`, which corresponds to $\alpha = 5/7$ for a local linear and $\alpha = 9/11$ for a local cubic regression, `InfR = "Nai"` with $\alpha = 5/9$ and $\alpha = 9/13$ for local linear and local cubic regressions, respectively, and `InfR = "Var"`, which always sets $\alpha = 1/2$.
- `bStart` is the (relative) starting bandwidth for the bandwidth selection algorithm. The default is `bStart = 0.15`. Note that $\hat{h}$ is usually not affected by the initial value.
- With the argument `bvc` the estimation of c_f can be adjusted even further. By setting `bvc = "Y"`, the bandwidth for estimating c_f will be enlarged by the factor CF as in (8). CF is omitted for `bvc = "N"`.
- `bb` describes the boundary method. If `bb = 0` is selected, the number of observations considered for smoothing is decreasing towards the boundaries, while it is constant throughout for `bb = 1`.
- `cb` is the proportion of observations at each boundary that is omitted in the bandwidth selection process.
- Via `method` the final smoothing method, after the bandwidth has been selected, can be defined. The default `method = "lpr"` corresponds to local polynomial estimates, whereas with `method = "kr"` a kernel regression is conducted.

A simplified version of `tsmooth()` for estimating a time series trend with a data-driven local polynomial is

```
msmooth(y, p, mu, bStart, alg, method),
```

which shares most of its arguments with `tsmooth()`. The only unknown argument is `alg`.

- `alg` defines specific argument settings of `tsmooth()` as named subalgorithms. The two algorithms **AlgA** and **AlgB** described in the section *The IPI-algorithm for estimating m* can be directly chosen by `alg = "A"` and `alg = "B"`, respectively.

In accordance with Feng et al. (2020), we propose the use of **AlgA** with the optimal inflation factor for local linear regression. For local cubic regression with a moderate n, **AlgB** with the stronger inflation factor should be used. If n is big enough, e.g. bigger than 400, the combination of $p = 3$ and **AlgA** will also lead to suitable results. In case when slight over smoothing is wished/required, the inflation factor `"Nai"` or even `"Var"` should be employed.

To estimate the first or second derivative of a time series trend with a data-driven local polynomial, the function

```
dsmooth(y, d, mu, pp, bStart.p, bStart)
```

should be employed. With this function, the first and second derivatives are always estimated using a local quadratic and local cubic regression, respectively. To obtain $\hat{c}_f$, a data-driven pilot estimate of the trend function via `tsmooth()` is computed.

- `d` specifies the order of derivative of the trend that will be estimated. Currently, `d = 1` and `d = 2` are valid options.
- `pp` corresponds to the order of polynomial considered for the pilot estimate of the trend function. `pp = 1` and `pp = 3` are available.
- `bStart.p` is the starting bandwidth for the data-driven pilot estimate of the trend.

Note that here, `bStart` is the initial bandwidth considered in the bandwidth selection for the derivative series. For further details on the functions we refer the reader to the user's guideline for this package. Beside the above functions, two S3 methods are also built in the package: a `print()` and a `plot()` method for objects of class `"smoots"`, a newly introduced class of objects created by the **smoots** package. They allow for a quick and detailed overview of the estimation results. The `"smoots"` objects themselves are generally lists consisting of different components such as input data and estimation results.

5 Simple application of smoots

In this and the next two sections, the wide applicability of the above mentioned functions will be illustrated by four real data examples. Those datasets are built in the package so that the reader can also use them. They are: `tempNH`, `gdpUS`, `dax` and `vix`, which contain observations of the mean monthly temperature changes of the Northern Hemisphere, the US GDP and daily financial data of the German stock index (DAX) and the CBOE Volatility Index (VIX), respectively. For further information see also the documentation on the data within the **smoots** package. Since the package is available on CRAN, the commands `install.packages("smoots")` and `library(smoots)` can be used to install it and attach it within the current R environment.

Direct application of the Semi-ARMA

To show the application of the additive Semi-ARMA model defined by (1) and (6), the time series of the mean monthly Northern Hemisphere temperature changes from 1880 to 2018 (NHTM) is chosen. The data are downloaded from the website of the NASA. For this purpose, the function `tsmooth()` is applied. The used settings of the arguments for this function are equivalent to those in algorithm A with $p = 1$.

```
tempChange <- smoots::tempNH$Change
est_temp <- smoots::tsmooth(tempChange, p = 1, mu = 2, Mcf = "NP", InfR = "Opt",
  bStart = 0.1, bvc = "Y", method = "lpr")
d1_temp <- smoots::dsmooth(tempChange, d = 1, mu = 2, pp = 3, bStart.p = 0.2,
  bStart = 0.15)
d2_temp <- smoots::dsmooth(tempChange, d = 2, mu = 3, pp = 1, bStart.p = 0.1,
  bStart = 0.2)
```

Figure 1(a) shows the observations together with the estimated trend. Here, the selected bandwidth is 0.1221. We see, the estimated trend fits the data very well. In particular, the trend increases steadily after 1970 that might be a signal for possible global warming in the last decades. The residuals are displayed in Figure 1(b), which look quite stationary. This indicates that Model (1) is a suitable approach for this time series. Moreover, Figures 1(c) and 1(d) illustrate the data-driven estimates of the first and second derivatives of the trend respectively, which fit the features of the trend function very well and provide us further details about the global temperature changes.

```
arma1 <- stats::arima(est_temp$res, order = c(1, 0, 1), include.mean = FALSE)
```

Consequently, an ARMA(1, 1) model is fitted to the residual series $\tilde{\xi}_t = y_t - \hat{m}(x_t)$ using the `arima()` function in R, which results in

$$\tilde{\xi}_t = 0.7489\tilde{\xi}_{t-1} - 0.3332\varepsilon_{t-1} + \varepsilon_t. \tag{10}$$

We see, the dependence of errors is dominated by a strong positive AR parameter with a moderate negative MA coefficient.

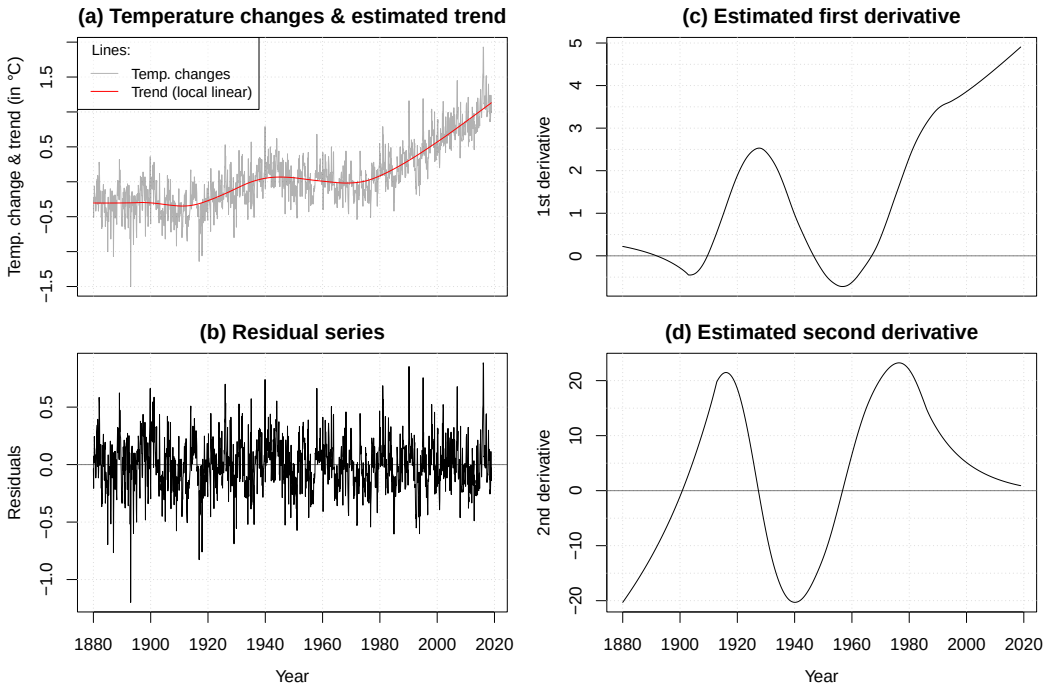


Figure 1: The NHTM series and the estimated trend are displayed in (a). The residuals, as well as the estimated first and second derivatives are shown in (b), (c) and (d), respectively.

A semiparametric log-local-linear growth model

A well-known approach in developing economics is the log-linear growth model. Assume that the log-transformation of a macroeconomic time series follows Models (1) and (6), we achieve a semiparametric local polynomial, in particular a local-linear extension of this theory. To show this application, the series of the quarterly US-GDP from the first quarter of 1947 to the second quarter of 2019, downloaded from the Federal Reserve Bank of St. Louis, is chosen. Data-driven local linear regression is applied to estimate the trend from the log-data using *AlgA* with the selected bandwidth 0.1325. A kernel regression estimate using the same bandwidth is also carried out for comparison.

```
l_gdp <- log(smoots::gdpUS$GDP)
gdp_t1 <- smoots::msmooth(l_gdp, p = 1, mu = 1, bStart = 0.1, alg = "A",
  method = "lpr")
gdp_t2 <- smoots::msmooth(l_gdp, p = 1, mu = 1, bStart = 0.1, alg = "A",
  method = "kr")
gdp_d1 <- smoots::dsmooth(l_gdp, d = 1, mu = 1, pp = 1, bStart.p = 0.1,
  bStart = 0.15)
gdp_d2 <- smoots::dsmooth(l_gdp, d = 2, mu = 1, pp = 1, bStart.p = 0.1,
  bStart = 0.2)
```

The results together with the log-data are displayed in Figure 2(a). We see that the two trend estimates in the middle part coincide with each other. They differ from each other only at the boundary points and the kernel estimate is affected by a clear boundary problem. Thus, the local linear method should be used. Residuals of this estimate are shown in Figure 2(b). Again, the estimated first and second derivatives are given in Figures 2(c) and 2(d), respectively, which help us to discover more detailed features of the economic development in the US.

```
arma2 <- stats::arima(gdp_t1$res, order = c(1, 0, 1), include.mean = FALSE)
```

Furthermore, the following ARMA(1, 1) model is obtained from the residuals:

$$\tilde{\xi}_t = 0.9079\tilde{\xi}_{t-1} + 0.2771\varepsilon_{t-1} + \varepsilon_t. \tag{11}$$

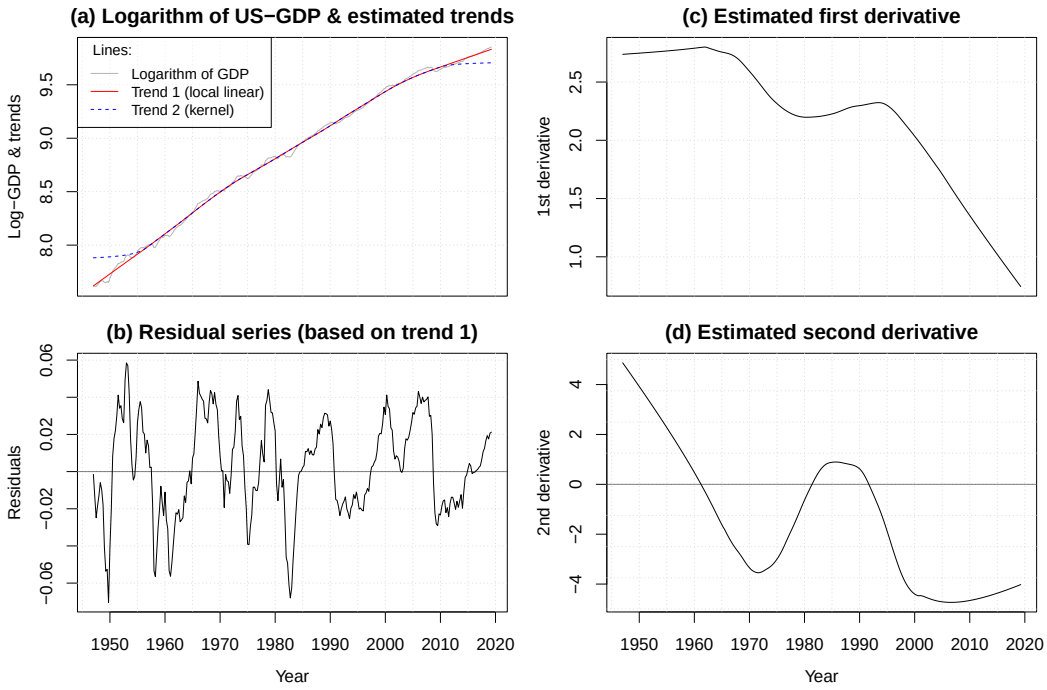


Figure 2: The log-transformed GDP series and the estimated trends are displayed in (a). The residuals based on the estimated local linear trend, as well as the estimated first and second derivatives are shown in (b), (c) and (d), respectively.

6 The Semi-Log-GARCH model

Most of the GARCH extensions, including the Log-GARCH (Pantula, 1986; Geweke, 1986; Milhøj, 1987), are defined for stationary return series. Recently, Francq et al. (2013) rediscovered the usefulness of the Log-GARCH. In practice it is however found that the unconditional variance of financial returns may change slowly over time and are hence non-stationary. To overcome this problem a semiparametric GARCH (Semi-GARCH) approach is proposed by Feng (2004), by defining the return series as a product of a (deterministic) smooth scale function and a GARCH model. Another well-known closely related approach is the Spline-GARCH introduced by Engle and Rangel (2008). In this paper we will introduce a Semi-Log-GARCH (semiparametric Log-GARCH), defined as a Log-GARCH with a smooth scale function. We propose to estimate the scale function in the Semi-Log-GARCH model based on the log-transformation of the squared returns as proposed by Engle and Rangel (2008).

Denote the centralized returns by r_t, $t = 1, ..., n$. The Semi-Log-GARCH model is defined by

$$r_t = \sqrt{v(x_t)}\zeta_t \text{ with } \zeta_t = \sqrt{h_t}\eta_t \text{ and} \tag{12}$$

$$\ln(h_t) = \omega + \sum_{i=1}^{l} \alpha_i \ln\left(\zeta_{t-i}^2\right) + \sum_{j=1}^{s} \beta_j \ln\left(h_{t-j}\right), \tag{13}$$

where $v(x_t) > 0$ is a smooth local variance component, $h_t > 0$ are conditional variances and η_t are i.i.d. random variables with zero mean and unit variance. It is assumed that ζ_t also has unit variance and $\zeta_t \neq 0$ almost surely, so that the model is well-defined. Let $y_t = \ln\left(r_t^2\right)$, $\xi_t = \ln\left(\zeta_t^2\right) - \mu_{lz}$ and $m(x_t) = \ln[v(x_t)] + \mu_{lz}$, where $\mu_{lz} = E\left[\ln\left(\zeta_t^2\right)\right]$. We see that the log-transformation of r_t^2 of the Semi-Log-GARCH model has the form $y_t = m(x_t) + \xi_t$, which is a special case of Model (1). Furthermore, define $\varepsilon_t = \ln\left(\eta_t^2\right) - \mu_{le}$ with $\mu_{le} = E\left[\ln\left(\eta_t^2\right)\right]$, which are i.i.d. zero mean innovations in the stationary process ξ_t. According to Francq and Sucarrat (2018), ξ_t has the following ARMA representation:

$$\xi_t = \sum_{i=1}^{l^*} \varphi_i \xi_{t-i} + \sum_{j=1}^{s} \psi_j \varepsilon_{t-j} + \varepsilon_t \tag{14}$$

with $l^* = \max(l, s)$. Thus, the Semi-Log-GARCH model is equivalent to a Semi-ARMA of the log-transformation of r_t^2 with the restriction that the AR order should not be less than the MA order. Hence, this model can be simply estimated using the **smoots** package and the `arima()` function of the **stats**

package. To obtain the total volatilities $\sigma_t = \sqrt{v(x_t)\, h_t}$, we suggest to utilize the three-step estimation procedure as proposed in Section 3.3 of Sucarrat (2019), however, we recommend to explicitly conduct the auxiliary regression in Step 1 in Sucarrat (2019) by using a data-driven local polynomial.

i) Obtain estimates of the nonparametric trend function $\hat{m}(x_t)$ in y_t via the **smoots** package.

ii) Fit an ARMA(l^*, s) model (14) to the residuals $\tilde{\xi}_t = y_t - \hat{m}(x_t)$, for example with the `arima()` function of the **stats** package, and compute the ARMA residuals $\hat{\varepsilon}_t$.

iii) Consider $\hat{\mu}_{le} = -\ln\left[\frac{1}{n}\sum_{i=1}^{n}\exp(\hat{\varepsilon}_i)\right]$ as an estimator of the log-moment $E\left[\ln\left(\eta_t^2\right)\right]$. Then the total volatilities are estimated as

$$\hat{\sigma}_t = \exp\left\{\left[\tilde{\xi}_t - \hat{\varepsilon}_t + \hat{m}(x_t) - \hat{\mu}_{le}\right]/2\right\}. \tag{15}$$

Estimates of the conditional volatilities are also calculable similarly by following the same three-step procedure while replacing (15) with $\sqrt{\hat{h}_t} = \exp\left[\left(\tilde{\xi}_t - \hat{\varepsilon}_t + \hat{\mu}_{lz} - \hat{\mu}_{le}\right)/2\right]$, where we suggest $\hat{\mu}_{lz} = -\ln\left[\frac{1}{n}\sum_{i=1}^{n}\exp\left(\tilde{\xi}_i\right)\right]$. An important characteristic of the illustrated estimation procedure is that explicit estimates of the Log-GARCH parameters ω, α_i, for $1 \leq i \leq p$, and β_j, for $1 \leq j \leq q$, are not required for computing $\hat{\sigma}_t$ or $\sqrt{\hat{h}_t}$ (Sucarrat, 2019). If, however, necessary, estimates could be derived by considering the relationships between the coefficients in (13) and (14), which are $\alpha_i = \varphi_i + \psi_i$, $\beta_j = -\psi_j$, where the non-existing coefficients are assumed to be zero, and $\omega = \left(1 - \sum_{i=1}^{l^*}\varphi_i\right)\mu_{lz} - \left(1 + \sum_{j=1}^{s}\psi_j\right)\mu_{le}$. Note that the parametric part of the Semi-Log-GARCH can also be estimated directly using the R package **lgarch** (Sucarrat, 2015). Nonetheless, this approach will not be considered in the current paper.

In the following, the DAX series from 1990 to July 2019 downloaded from Yahoo Finance is chosen to show the application of the Semi-Log-GARCH model. Note that an observed return can be sometimes exactly zero. To overcome this problem, the log-transformation is calculated for the squared centralized returns, which are a.s. non-zero. This would even be a necessary treatment, if the returns had a very small, but non-zero mean.

```
# Calculate the centralized log-returns
dax_close <- smoots::dax$Close; dax <- diff(log(dax_close))
rt <- dax - mean(dax); yt <- log(rt ^ 2)
```

Subsequently, the previously described estimation procedure according to Sucarrat (2019) is implemented by estimating the trend function in the logarithm of the squared returns `yt` with the function `msmooth()` of the **smoots** package. More specifically, a local cubic trend is fitted, while employing *AlgA*.

```
# Step 1: Estimate the trend in the log-transformed data using 'smoots'
estim3 <- smoots::msmooth(yt, p = 3, alg = "A")
m_xt <- estim3$ye

# Step 2: Fit an ARMA model to the residuals
xi <- estim3$res
arma3 <- arima(xi, order = c(1, 0, 1), include.mean = FALSE)

# Step 3: Estimate further quantities and the total volatilities
mu_le <- -log(mean(exp(arma3$residuals)))
vol <- exp((xi - arma3$residuals + m_xt - mu_le) / 2)
```

For reference, `estim3.2 <- smoots::msmooth(yt, p = 1, alg = "A")` is called, i.e. a local linear trend under consideration of *AlgA* is estimated as well. The centralized log-returns and the log-transformed data with the two estimated trends (local linear: blue; local cubic: red) are displayed in Figures 3(a) and 3(b). Moreover, the selected bandwidths are 0.0869 and 0.1013, respectively. Results in Figure 3(b) indicate that the unconditional variance of the DAX-returns changes slowly over time. Ultimately, the local cubic trend is chosen for further analysis, because here the results of the local linear approach are over-smoothed. The following ARMA(1, 1) model (see Step 2 in the code) is obtained from the residuals of the log-data

$$\tilde{\xi}_t = 0.9692\tilde{\xi}_{t-1} - 0.9221\varepsilon_{t-1} + \varepsilon_t, \tag{16}$$

whereas the re-transformed Log-GARCH(1, 1) formula is given by

$$\ln(h_t) = 0.0685 + 0.0471\ln\left(\zeta_{t-1}^2\right) + 0.9221\ln(h_{t-1}) \tag{17}$$

following the idea of Sucarrat et al. (2016). The estimated conditional volatility ($\sqrt{\hat{h}_t}$) and total volatility ($\hat{\sigma}_t$) series are displayed in Figures 3(c) and 3(d).

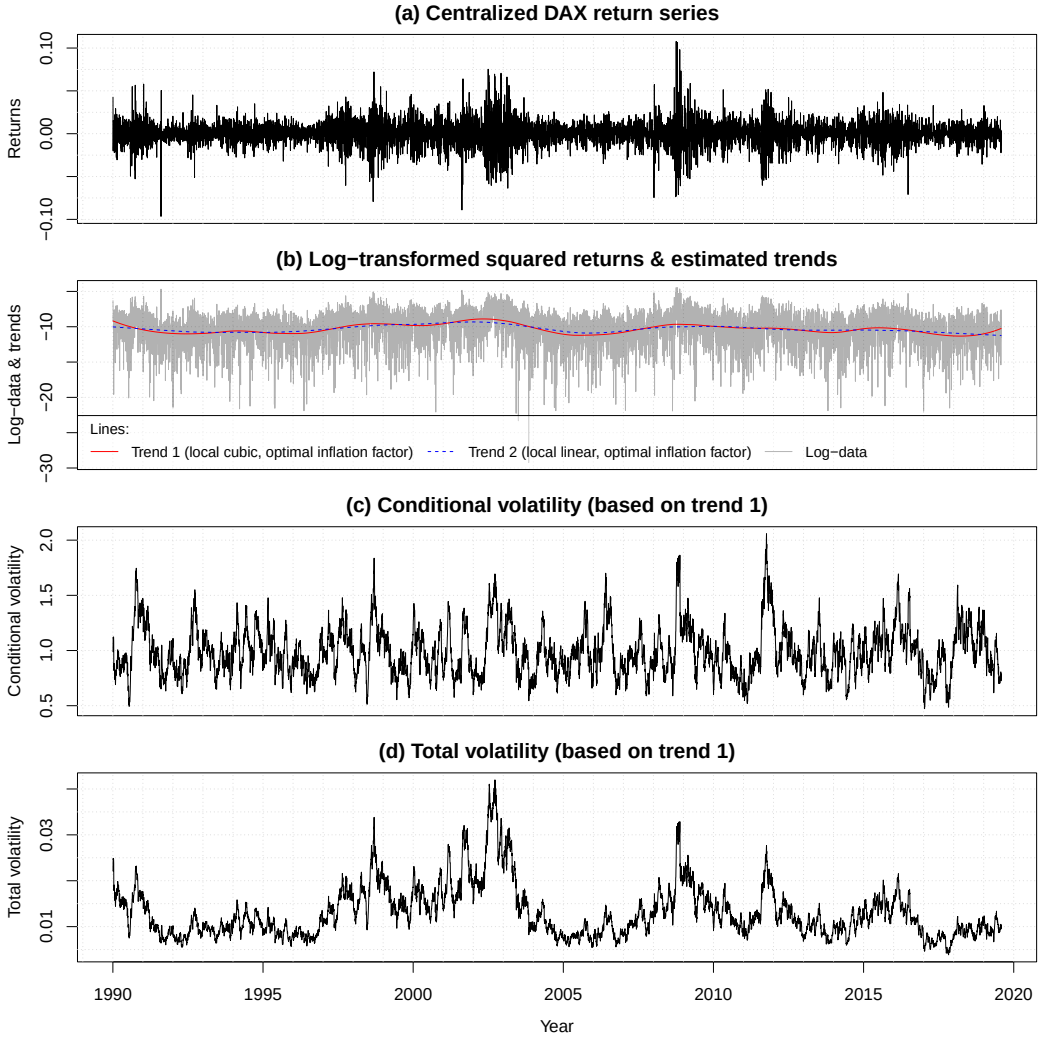


Figure 3: The demeaned DAX returns are displayed in (a). The log-transformed squared returns and the estimated trends are shown in (b). Based on the estimated local cubic trend, the corresponding estimated conditional and total volatilities are illustrated in (c) and (d), respectively.

7 The Semi-Log-ACD model

A more general framework for modeling non-negative financial time series is the ACD (autoregressive conditional duration, Engle and Russell, 1998) model, which corresponds to a squared GARCH model and can be applied to both high-frequency or daily financial data. Logarithmic extensions of this approach were introduced by Bauwens and Giot (2000), where the Type I definition (called a Log-ACD) corresponds to a squared form of the Log-GARCH. Semiparametric generalization of the Log-ACD (Semi-Log-ACD) was defined and applied to different kinds of non-negative financial data by Forstinger (2018). In this paper, the application of the Semi-Log-ACD model will be illustrated by the CBOE Volatility Index (VIX) from 1990 to July 2019, denoted by V_t, $t = 1, ..., n$. The data was again downloaded from Yahoo Finance. The Semi-Log-ACD model for V_t is defined by

$$V_t = g(x_t)\,\lambda_t e_t, \tag{18}$$

where $u_t = \lambda_t e_t$ follows a Log-ACD and $g \geq 0$ is a smooth mean function in V_t, $\lambda_t \geq 0$ is the conditional mean and e_t is an i.i.d. series of non-negative random variables. It is assumed that $E(\lambda_t) = E(e_t) = 1$. Further investigation on this model can be carried out similarly to that on the Semi-Log-GARCH model by replacing r_t^2, h_t and η_t^2 there with V_t, λ_t and e_t, respectively. The Semi-Log-ACD model can be similarly estimated. Discussion of those details is omitted. For further information we refer the reader to Forstinger (2018) and references therein.

```
# Calculate the logarithm of the index
V <- smoots::vix$Close; lnV <- log(V)
```

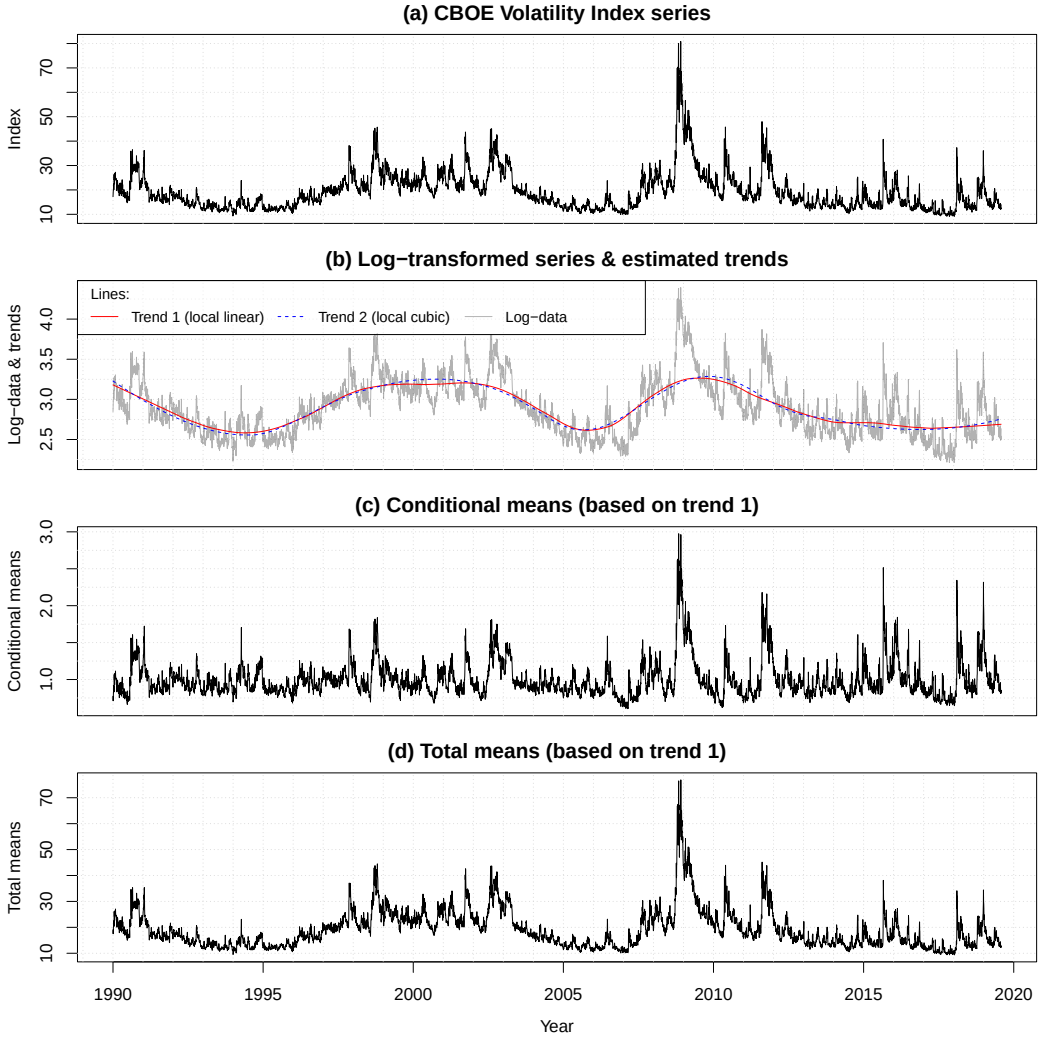


Figure 4: The VIX series is displayed in (a). The log-transformed index and the estimated trends are shown in (b). Based on the estimated local linear trend, the corresponding estimated conditional and total means are illustrated in (c) and (d), respectively.

```
# Step 1: Estimate the trend in the log-transformed data using 'smoots'
estim4 <- smoots::msmooth(lnV)
estim4.2 <- smoots::msmooth(lnV, p = 3, alg = "B")
m_xt <- estim4$ye

# Step 2: Fit an ARMA model to the residuals
xi <- estim4$res
arma4 <- arima(xi, order = c(1, 0, 1), include.mean = FALSE)

# Step 3: Estimate further quantities and the total means
mu_le <- -log(mean(exp(arma4$residuals)))
means <- exp(xi - arma4$residuals + m_xt - mu_le)
```

The original series of V_t is shown in Figure 4(a). The trend was estimated from the log-transformation of V_t using both *AlgA* and *AlgB* with the selected bandwidths 0.0771 and 0.1598, respectively. The data (black), the local linear trend (red) and the local cubic trend (blue) are displayed in Figure 4(b). The results using both algorithms are quite similar, except the local cubic estimates are smoother. From the residuals of the local linear approach the following ARMA(1, 1) model (see Step 2 in the code) is obtained:

$$\tilde{\xi}_t = 0.9626\tilde{\xi}_{t-1} - 0.0707\varepsilon_{t-1} + \varepsilon_t. \tag{19}$$

Subsequently, the estimated log-form of the conditional mean function is given by

$$\ln(\lambda_t) = 0.0010 + 0.8919\ln(u_{t-1}) + 0.0707\ln(\lambda_{t-1}). \tag{20}$$

The estimated conditional means and the total means in the original data by the local linear approach are shown in Figures 4(c) and 4(d). We see the results fit the data very well. This model can be applied for forecasting the VIX in the future.

8 Concluding remarks

In this paper the methodological background for developing the R package **smoots** (version 1.0.1) is summarized. The usage of the main functions in this package is explained in detail. To show the wide applicability of this approach two new semiparametric models for analyzing financial time series are also introduced. Data examples show that the proposed approach can be applied to different kinds non-stationary time series and the developed R package works very well for data-driven implementation of those semiparametric time series models. In particular, non-negative time series following a semiparametric multiplicative model can be easily estimated via the log-transformation. It is found that the errors in some examples could exhibit clear long memory. However, the current package is developed under short memory assumption. It is hence worthy to study the possible extension of the current approach to semiparametric time series models with long memory errors. Further extensions of the proposals in this paper, such as the development of suitable forecasting procedures and tools for testing stationarity of the errors or linearity of the deterministic trend, should also be studied in the future and included in future versions of the **smoots** package.

9 Computational details

The numerical results in this paper were obtained using R 4.1.1 with the **smoots** 1.0.1 package and the **stats** 4.1.1 package. R itself and all packages used are available from CRAN at https://CRAN.R-project.org/.

Acknowledgments: This work was supported by the German DFG project GZ-FE-1500-2-1. The data used were downloaded from different public sources as indicated in the contexts. We are grateful to the CRAN-network for great help during the publication of the R package **smoots**. We would also like to thank Dr. Marlon Fritz for helpful discussions.

Bibliography

D. M. Allen. The relationship between variable selection and data agumentation and a method for prediction. *Technometrics*, 16(1):125–127, 1974. URL https://doi.org/10.1080/00401706.1974.10489157. [p182]

N. S. Altman. Kernel smoothing of data with correlated errors. *Journal of the American Statistical Association*, 85(411):749–759, 1990. URL https://doi.org/10.1080/01621459.1990.10474936. [p182]

M. Balcilar. *mFilter: Miscellaneous Time Series Filters*, 2019. URL https://CRAN.R-project.org/package=mFilter. R package version 0.1-5. [p182]

L. Bauwens and P. Giot. The logarithmic ACD model: An application to the bid-ask quote process of three NYSE stocks. *Annales d'Economie et de Statistique*, (60):117–149, 2000. URL https://doi.org/10.2307/20076257. [p183, 191]

M. Baxter and R. G. King. Measuring business cycles: Approximate band-pass filters for economic time series. *Review of Economics and Statistics*, 81(4):575–593, 1999. URL https://doi.org/10.1162/003465399558454. [p182]

J. Beran, Y. Feng, and S. Heiler. Modifying the double smoothing bandwidth selector in nonparametric regression. *Statistical Methodology*, 6(5):447–465, 2009. URL https://doi.org/10.1016/j.stamet.2009.04.001. [p182]

T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986. URL https://doi.org/10.1016/0304-4076(86)90063-1. [p183]

P. Bühlmann. Locally adaptive lag-window spectral estimation. *Journal of Time Series Analysis*, 17(3):247–270, 1996. URL https://doi.org/10.1111/j.1467-9892.1996.tb00275.x. [p182, 183, 184]

R. F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society*, 50(4):987–1007, 1982. URL https://doi.org/10.2307/1912773. [p183]

R. F. Engle and J. G. Rangel. The Spline-GARCH model for low-frequency volatility and its global macroeconomic causes. *The Review of Financial Studies*, 21(3):1187–1222, 2008. URL https://doi.org/10.2139/ssrn.939447. [p189]

R. F. Engle and J. R. Russell. Autoregressive conditional duration: A new model for irregularly spaced transaction data. *Econometrica*, 66(5):1127–1162, 1998. URL https://doi.org/10.2307/2999632. [p183, 191]

Y. Feng. Simultaneously modeling conditional heteroskedasticity and scale change. *Econometric Theory*, 20(3):563–596, 2004. URL https://doi.org/10.1017/S0266466604203061. [p189]

Y. Feng. On the asymptotic variance in nonparametric regression with fractional time-series errors. *Nonparametric Statistics*, 19(2):63–76, 2007. URL https://doi.org/10.1080/10485250701381737. [p182]

Y. Feng and S. Heiler. A simple bootstrap bandwidth selector for local polynomial fitting. *Journal of Statistical Computation and Simulation*, 79(12):1425–1439, 2009. URL https://doi.org/10.1080/00949650802352019. [p185]

Y. Feng and D. Schulz. *smoots: Nonparametric Estimation of the Trend and Its Derivatives in TS*, 2019. URL https://CRAN.R-project.org/package=smoots. R package version 1.0.1. [p182]

Y. Feng, T. Gries, and M. Fritz. Data-driven local polynomial for the trend and its derivatives in economic time series. *Journal of Nonparametric Statistics*, 32(2):510–533, 2020. URL https://doi.org/10.1080/10485252.2020.1759598. [p182, 183, 184, 185, 186]

S. Forstinger. *Modelling and Forecasting Financial and Economic Time Series Using Different Semiparametric ACD Models*. PhD thesis, Paderborn, Universität Paderborn, 2018. [p191]

M. Francisco-Fernández, J. Opsomer, and J. M. Vilar-Fernández. Plug-in bandwidth selector for local polynomial regression estimator with correlated errors. *Nonparametric Statistics*, 16(1-2):127–151, 2004. URL https://doi.org/10.1080/10485250310001622848. [p182]

C. Francq and G. Sucarrat. An exponential chi-squared QMLE for Log-GARCH models via the ARMA representation. *Journal of Financial Econometrics*, 16(1):129–154, 2018. URL https://doi.org/10.1093/jjfinec/nbx032. [p189]

C. Francq, O. Wintenberger, and J.-M. Zakoïan. GARCH models without positivity constraints: Exponential or Log GARCH? *Journal of Econometrics*, 177(1):34–46, 2013. URL https://doi.org/10.1016/j.jeconom.2013.05.004. [p183, 189]

T. Gasser, A. Kneip, and W. Köhler. A flexible and fast method for automatic smoothing. *Journal of the American Statistical Association*, 86(415):643–652, 1991. URL https://doi.org/10.1080/01621459.1991.10475090. [p182]

J. Geweke. Comment on: Modelling the persistence of conditional variances. *Econometric Reviews*, 5(1): 57–61, 1986. URL https://doi.org/10.1080/07474938608800097. [p183, 189]

J. D. Hart. Kernel regression estimation with time series errors. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(1):173–187, 1991. URL https://doi.org/10.1111/j.2517-6161.1991.tb01816.x. [p182]

E. Herrmann and M. Maechler. *lokern: Kernel Regression Smoothing with Local or Global Plug-in Bandwidth*, 2021. URL https://CRAN.R-project.org/package=lokern. R package version 1.1-9. [p182]

R. J. Hodrick and E. C. Prescott. Postwar US business cycles: An empirical investigation. *Journal of Money, Credit and Banking*, 29(1):1–16, 1997. URL https://doi.org/10.2307/2953682. [p182]

A. Milhøj. *A Multiplicative Parameterization of ARCH Models*. University of Copenhagen, Department of Statistics, 1987. [p183, 189]

H.-G. Müller. *Nonparametric Regression Analysis of Longitudinal Data*. Springer, New York, NY, 1988. URL https://doi.org/10.1007/978-1-4612-3926-0. [p185]

J. L. Ojeda Cabrera. *locpol: Kernel Local Polynomial Regression*, 2018. URL https://CRAN.R-project.org/package=locpol. R package version 0.7-0. [p182]

J. Opsomer. Nonparametric regression in the presence of correlated errors. In *Modelling Longitudinal and Spatially Correlated Data*, pages 339–348. Springer, New York, NY, 1997. URL https://doi.org/10.1007/978-1-4612-0699-6_30. [p182]

S. G. Pantula. Modeling the persistence of conditional variances: A comment. *Econometric Reviews*, 5 (1):71–74, 1986. URL https://doi.org/10.1080/07474938608800099. [p183, 189]

D. Qiu. *rmaf: Refined Moving Average Filter*, 2015. URL https://CRAN.R-project.org/package=rmaf. R package version 3.0.1. [p182]

D. Ruppert, S. J. Sheather, and M. P. Wand. An effective bandwidth selector for local least squares regression. *Journal of the American Statistical Association*, 90(432):1257–1270, 1995. URL https://doi.org/10.1080/01621459.1995.10476630. [p182]

G. Sucarrat. *lgarch: Simulation and Estimation of Log-GARCH Models*, 2015. URL https://CRAN.R-project.org/package=lgarch. R package version 0.6-2. [p190]

G. Sucarrat. The log-GARCH model via ARMA representations. In *Financial Mathematics, Volatility and Covariance Modelling*, volume 2, pages 336–359. Routledge, London, 1 edition, 2019. URL https://doi.org/10.4324/9781315162737. [p183, 190]

G. Sucarrat, S. Grønneberg, and A. Escribano. Estimation and inference in univariate and multivariate log-GARCH-X models when the conditional density is unknown. *Computational Statistics & Data Analysis*, 100:582–594, 2016. URL https://doi.org/10.1016/j.csda.2015.12.005. [p190]

M. Wand. *KernSmooth: Functions for Kernel Smoothing Supporting Wand & Jones (1995)*, 2021. URL https://CRAN.R-project.org/package=KernSmooth. R package version 2.23-20. [p182]

Yuanhua Feng
Department of Economics
Faculty of Business Administration and Economics
Paderborn University
Warburger Straße 100, 33098 Paderborn
Germany
yuanhua.feng@uni-paderborn.de

Thomas Gries
Department of Economics
Faculty of Business Administration and Economics
Paderborn University
Warburger Straße 100, 33098 Paderborn
Germany
thomas.gries@uni-paderborn.de

Sebastian Letmathe
Department of Economics
Faculty of Business Administration and Economics
Paderborn University
Warburger Straße 100, 33098 Paderborn
Germany
sebastian.letmathe@uni-paderborn.de

Dominik Schulz
Department of Economics
Faculty of Business Administration and Economics
Paderborn University
Warburger Straße 100, 33098 Paderborn
Germany
dominik.schulz@uni-paderborn.de

cpsurvsim: An R Package for Simulating Data from Change-Point Hazard Distributions

by Camille J. Hochheimer and Roy T. Sabo

Abstract Change-point hazard models have several practical applications, including modeling processes such as cancer mortality rates and disease progression. While the inverse cumulative distribution function (CDF) method is commonly used for simulating data, we demonstrate the shortcomings of this approach when simulating data from change-point hazard distributions with more than a scale parameter. We propose an alternative method of simulating this data that takes advantage of the memoryless property of survival data and introduce the R package **cpsurvsim** which implements both simulation methods. The functions of **cpsurvsim** are discussed, demonstrated, and compared.

1 Introduction: Simulating from time-to-event processes in R

When modeling time-to-event processes, especially over long periods of time, it is often unreasonable to assume a constant hazard rate. In these cases, change-point hazard models are applicable. The majority of research surrounding change-point hazard models focuses on the Cox proportional hazards and piecewise exponential models with one change-point (Yao, 1986; Gijbels and Gurler, 2003; Wu et al., 2003; Rojas et al., 2011; Dupuy, 2006), likely due to the straightforward extension for including fixed and time-varying covariates (Zhou, 2001; Hendry, 2014; Montez-Rath et al., 2017a; Wong et al., 2018). Research on hazard models with multiple change-points is also expanding as these models have a wide range of applications in fields such as medicine, public health, and economics (Liu et al., 2008; Goodman et al., 2011; He et al., 2013; Han et al., 2014; Qian and Zhang, 2014; Cai et al., 2017). In the interest of simulating time-to-event data featuring trends with multiple change-points, Walke (2010) presents an algorithm for simulating data from the piecewise exponential distribution with fixed type I censoring using the location of the change-points, the baseline hazard, and the relative hazard for each time interval in between change-points. As the research surrounding parametric change-point hazard models with multiple change-points continues to grow, likewise does the need to simulate data from these distributions. Simulation is also a powerful and popular tool for assessing the appropriateness of a model for one's data or conducting a power analysis.

Several R packages available from the Comprehensive R Archive Network (CRAN) provide functions for simulating time-to-event data in general, with a heavy focus on the Cox model. Some of the more popular packages are provided in Table 1, which expands on the METACRAN compilation (Allignol and Latouche, 2020). Although considerably smaller in scope, a few R packages provide functions for simulating data with change-points. **CPsurv** has functionality for simulating both nonparametric survival data and parametric survival data from the Weibull change-point distribution but requires existing data as an argument and only allows for one change-point (Krügel et al., 2017). **SimSCRPiecewise** simulates data using the piecewise exponential hazard model within the Bayesian framework, however, this method requires at least one covariate as an argument (Chapple, 2016).

Our package **cpsurvsim** allows users to simulate data from a both the exponential and Weibull hazard models with type I right censoring allowing for multiple change-points (Hochheimer, 2021). **cpsurvsim** provides two methods for simulating data, which are introduced in the following section. The first method draws on Walke (2010), using the inverse hazard function to simulate data. The second employs the memoryless simulation method, the details of which are also discussed in the next section. We then demonstrate how to simulate data using **cpsurvsim** and compare the performance of these methods through a simulation study with the motivation of enabling users to determine which method is best for their data.

Package Title	Brief Description
coxed	Simulates data for the Cox model using the flexible-hazard method and allows for the inclusion of time-varying covariates (Kropko and Harden, 2019)
CPsurv	Simulates one change-point for non-parametric survival analysis or parametric survival analysis using the Weibull distribution (Krügel et al., 2017)
cpsurvsim	Simulates data with multiple change-points from the exponential and Weibull distributions (Hochheimer, 2021)
discSurv	Simulates survival data from discrete competing risk models (Welchowski and Schmid, 2019)
gems	Simulates data from multistate models and allows for non-Markov models that account for previous events (Blaser et al., 2015)
genSurv	Gives users the option to generate data with a binary, time-dependent covariate (Araújo et al., 2015; Meira-Machado and Faria, 2014)
ipred	Provides a function for simulating survival data for tree-structured survival analysis (Peters and Hothorn, 2019)
MicSim	Performs continuous time miscrosimulations to simulate life courses (Zinn, 2018)
PermAlgo	Uses a permutational algorithm to generate time-to-event data allowing for the inclusion of several time-dependent covariates (Sylvestre et al., 2010)
prodlim	Has functions for simulating right censored non-parametric survival data with two covariates and with or without competing risks (Gerds, 2018)
simMSM	Uses inversion sampling to simulate data from multi-state models allowing for non-linear baseline hazards, time-varying covariates, and dependence on past events (Reulen, 2015)
simPH	Simulates data from Cox proportional hazards models (Gandrud, 2015)
simsurv	Simulates data from various parametric survival distributions, 2-component mixture distributions, and user-defined hazards (Brilleman, 2019)
SimSCRPiecewise	Uses Bayesian estimation to simulate data from the piecewise exponential hazard model allowing for the inclusion of covariates (Chapple, 2016)
SimulateCER	While not a formal R package, this package extends the methods found in **PermAlgo** and can be downloaded from GitHub (Montez-Rath et al., 2017b)
survsim	Allows users to simulate time-to-event, competing risks, multiple event, and recurrent event data (Moriña and Navarro, 2014)

Table 1: R packages for simulating time-to-event data

2 Simulating data from popular change-point hazard models

The piecewise exponential model with multiple change-points $(\tau_k, k = 1, \ldots, K)$ can be expressed as

$$f(t) = \begin{cases} \theta_1 \exp\{-\theta_1 t\} & 0 \leq t < \tau_1 \\ \theta_2 \exp\{-\theta_1 \tau_1 - \theta_2 (t - \tau_1)\} & \tau_1 \leq t < \tau_2 \\ \vdots & \\ \theta_{K+1} \exp\{-\theta_1 \tau_1 - \theta_2 (\tau_2 - \tau_1) - \ldots - \theta_{K+1}(t - \tau_K)\} & t \geq \tau_K \end{cases} \tag{1}$$

with corresponding hazard function

$$h(t) = \begin{cases} \theta_1 & 0 \leq t < \tau_1 \\ \theta_2 & \tau_1 \leq t < \tau_2 \\ \vdots & \vdots \\ \theta_{K+1} & t \geq \tau_K. \end{cases} \tag{2}$$

We draw on the work of Walke (2010) in that we use the inverse hazard function to simulate survival time t. Walke (2010) uses a baseline hazard and relative hazards to simulate each time interval between change-points, whereas our simulation is based on the value of the scale parameter $(\theta_i, i = 1, \ldots, K+1)$ corresponding to each interval as specified by the user. Starting with the relationship between the cumulative density function (CDF) and the cumulative hazard function $(F(t) = 1 - \exp(-H(t))$ where $H(t) = \int h(t)dt)$ and noting that $F(t) = U$ where U is a uniform random variable on (0,1), we derive $t = H^{-1}(-\log(1 - U))$. Seeing as $x = -\log(1 - U) \sim Exp(1)$, we can simulate random variables from the exponential distribution and plug them into the inverse hazard function to get simulated event time t. With this in mind, the inverse cumulative hazard function for the exponential change-point hazard model with four change-points is

$$H^{-1}(x) = \begin{cases} \frac{x}{\theta_1} & 0 \leq x < A \\ \frac{x-A}{\theta_2} + \tau_1 & A \leq x < A + B \\ \frac{x-A-B}{\theta_3} + \tau_2 & A + B \leq x < A + B + C \\ \frac{x-A-B-C}{\theta_4} + \tau_3 & A + B + C \leq x < A + B + C + D \\ \frac{x-A-B-C-D}{\theta_5} + \tau_4 & x \geq A + B + C + D \end{cases} \tag{3}$$

where $A = \theta_1 \tau_1$, $B = \theta_2(\tau_2 - \tau_1)$, $C = \theta_3(\tau_3 - \tau_2)$, and $D = \theta_4(\tau_4 - \tau_3)$. In **cpsurvsim**, this method of simulating time-to-event data is considered the CDF method. An end-of-study time horizon (or maximum measurement time) is specified by the user and all simulated event times with values greater than the end time are censored at that point (type I right censoring).

The Weibull distribution is another popular parametric model for survival data due to its flexibility to fit a variety of hazard shapes while still satisfying the proportional hazards assumption. Note that when $\gamma = 1$, it is identical to the exponential distribution. The Weibull change-point model has the probability density function

$$f(t) = \begin{cases} \theta_1 t^{\gamma-1} \exp\{-\frac{\theta_1}{\gamma} t^\gamma\} & 0 \leq t < \tau_1 \\ \theta_2 t^{\gamma-1} \exp\{-\frac{\theta_2}{\gamma}(t^\gamma - \tau_1^\gamma) - \frac{\theta_1}{\gamma}\tau_1^\gamma\} & \tau_1 \leq t < \tau_2 \\ \vdots & \\ \theta_{K+1} t^{\gamma-1} \exp\{-\frac{\theta_{K+1}}{\gamma}(t^\gamma - \tau_K^\gamma) - \frac{\theta_K}{\gamma}(\tau_K^\gamma - \tau_{K-1}^\gamma) - \ldots - \frac{\theta_1}{\gamma}\tau_1^\gamma\} & t \geq \tau_K \end{cases} \tag{4}$$

with corresponding hazard function

$$h(t) = \begin{cases} \theta_1 t^{\gamma-1} & 0 \leq t < \tau_1 \\ \theta_2 t^{\gamma-1} & \tau_1 \leq t < \tau_2 \\ \vdots & \vdots \\ \theta_{K+1} t^{\gamma-1} & t \geq \tau_K. \end{cases} \tag{5}$$

As with the exponential model, event times can be simulated using the inverse hazard function (shown

Function	Hazard model	Simulation method
`exp_cdfsim`	Piecewise constant	Inverse hazard function
`exp_memsim`	Piecewise constant	Memoryless
`weib_cdfsim`	Weibull change-point	Inverse hazard function
`weib_memsim`	Weibull change-point	Memoryless

Table 2: Summary of functions for simulating data using **cpsurvsim**

here with four change-points)

$$H^{-1}(x) = \begin{cases} (\frac{\gamma}{\theta_1}x)^{1/\gamma} & 0 \leq x < A \\ [\frac{\gamma}{\theta_2}(x-A)+\tau_1^\gamma]^{1/\gamma} & A \leq x < A+B \\ [\frac{\gamma}{\theta_3}(x-A-B)+\tau_2^\gamma]^{1/\gamma} & A+B \leq x < A+B+C \\ [\frac{\gamma}{\theta_4}(x-A-B-C)+\tau_3^\gamma]^{1/\gamma} & A+B+C \leq x < A+B+C+D \\ [\frac{\gamma}{\theta_5}(x-A-B-C-D)+\tau_4^\gamma]^{1/\gamma} & x \geq A+B+C+D \end{cases} \quad (6)$$

where $A = \frac{\theta_1}{\gamma}\tau_1^\gamma$, $B = \frac{\theta_2}{\gamma}(\tau_2^\gamma - \tau_1^\gamma)$, $C = \frac{\theta_3}{\gamma}(\tau_3^\gamma - \tau_2^\gamma)$, and $D = \frac{\theta_4}{\gamma}(\tau_4^\gamma - \tau_3^\gamma)$.

Zhou (2001) touches on the idea of the memoryless property as a means of interpreting the piecewise exponential model, however, we take this one step further by using this property to simulate data from change-point hazard models. In survival analysis, the memoryless property states that the probability of an individual experiencing an event at time t is independent of the probability of experiencing an event up to that point. Likewise, the probability of an event occurring after a change-point is independent of the probability that the event occurs before the change-point.

Our memoryless simulation method uses this extension of the memoryless property in that data between change-points are simulated from independent exponential or Weibull hazard distributions with scale parameters θ_i corresponding to each time interval. Participants with simulated survival times past the next change-point are considered surviving at least to that change-point and then an additional survival time is simulated for them in the next time interval. Total time to event is calculated as the sum of time in each interval between change-points, with those surviving past the study end time censored at that point. Survival times within each interval are calculated using the inverse hazard of the independent exponential or Weibull function representing that time period. In this way, the inverse hazard and memoryless methods are equivalent when there are no change-points.

3 The cpsurvsim package

The **cpsurvsim** package can be installed from CRAN. Functions for simulating data are summarized in Table 2

As an example of the functions `exp_cdfsim` and `weib_cdfsim`, which simulate data using the inverse hazard method from the exponential and Weibull distributions, respectively, consider the following:

```
library(cpsurvsim)
dta1 <- exp_cdfsim(n = 50, endtime = 100, theta = c(0.005, 0.01, 0.05),
+  tau = c(33, 66))
head(dta1)

       time censor
1 100.00000      0
2  85.99736      1
3  78.21772      1
4  71.03138      1
5 100.00000      0
6  82.71520      1

dta2 <- weib_cdfsim(n = 50, endtime = 100, gamma = 2,
+  theta = c(0.0001, 0.0002, 0.0001), tau = c(33, 66))
head(dta2)
```

```
       time censor
1  11.36844      1
2 100.00000      0
3  81.04904      1
4 100.00000      0
5  71.93590      1
6  56.40275      1
```

When simulating using the memoryless method, we use the following calls from **cpsurvsim**:

```
dta3 <- exp_memsim(n = 50, endtime = 100, theta = c(0.005, 0.01, 0.05),
+  tau = c(33, 66))
head(dta3)
```

```
      time censor
1 93.64262      1
2 63.47413      1
3 84.54253      1
4 89.01574      1
5 73.92685      1
6 23.67631      1
```

```
dta4 <- weib_memsim(n = 50, endtime = 100, gamma = 2,
+  theta = c(0.0001, 0.0002, 0.0001), tau = c(33, 66))
head(dta4)
```

```
       time censor
1  59.47848      1
2 100.00000      0
3  62.08739      1
4 100.00000      0
5 100.00000      0
6 100.00000      0
```

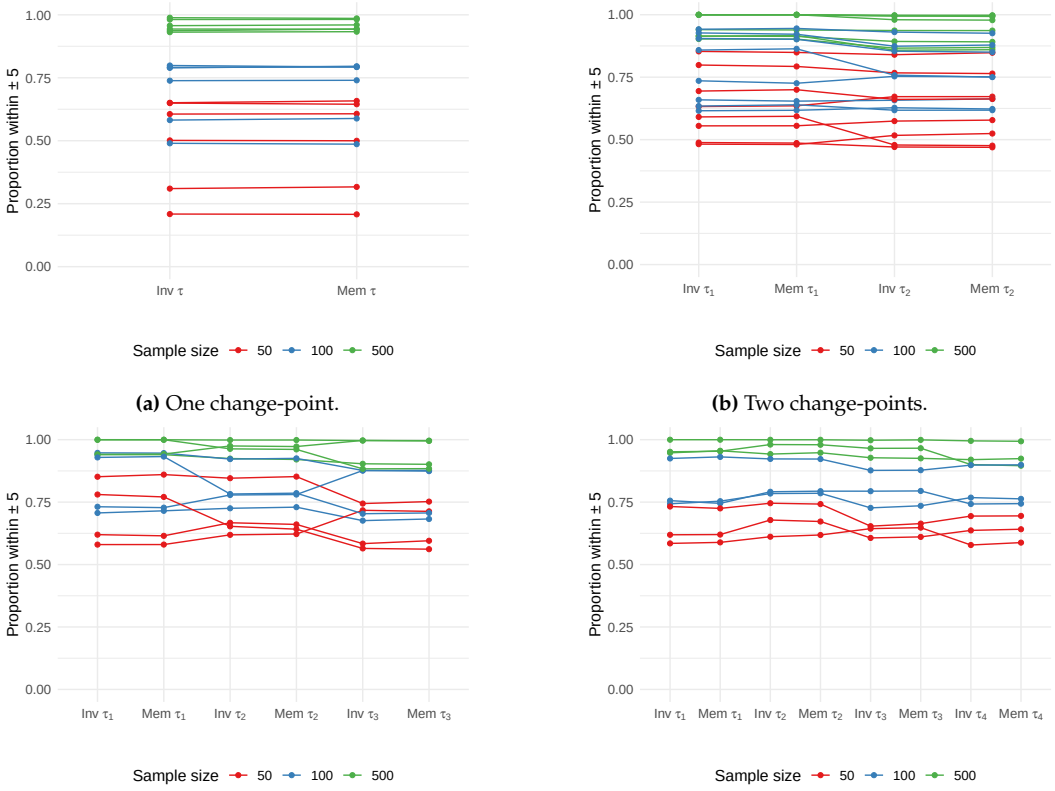
As seen in these examples, all four functions return a dataset with the survival times and a censoring indicator.

4 Comparison of simulation methods

To compare the performance of the inverse hazard method with the memoryless method under different settings, we conducted a simulation study using **cpsurvsim**. We simulated data with one, two, three, and four change-points using both the exponential and Weibull distributions. In our simulation, time t ranged from 0-100 and change-points occurred at various times within that range. Sample sizes of 50, 100, and 500 were tested and values of θ were chosen to demonstrate differences between the simulation methods when the hazard rate changes (e.g., smaller to larger hazard versus larger to smaller hazard). For the Weibull simulations, we set $\gamma = 2$. We conducted 10,000 simulations of each setting. We are primarily interested in comparing the ability of these two simulation methods to simulate data with the correct change-points τ_i. Therefore, we compared how often the estimated value ($\hat{\tau}_i$) was within a 10% range, in this case $[\tau_i - 5, \tau_i + 5]$ based on our time range. We also evaluated whether the known values of τ_i fell within the 95% confidence interval of the average simulated values for both methods as well as discuss bias in the model parameters. This simulation study was conducted in R 3.6.1.

When simulating from the exponential distribution, these two simulation methods had comparable accuracy in terms of the location of the change-points (see Figure 1). Sample size, however, had a large impact on the accuracy regardless of the simulation method. When estimating one change-point with a sample size of 50, there were a few simulation templates where less than a third of estimates $\hat{\tau}$ were within range of the known change-point (Figure 1a). In general, accuracy improved as the sample size increased. For every simulation scenario using the exponential distribution, the 95% confidence interval for the mean estimate of τ_i included the known value.

Simulations for the Weibull distribution, however, revealed important differences in accuracy between the two methods (see Figure 2). Although accuracy of the two methods was similar when simulating one change-point, there were many cases where accuracy was very low, even with a sample size of 500 (Figure 2a). In almost one quarter (4/18) of the simulation scenarios, the true value of τ was not within the 95% confidence interval of the average estimate for either method. In three of



(a) One change-point.

(b) Two change-points.

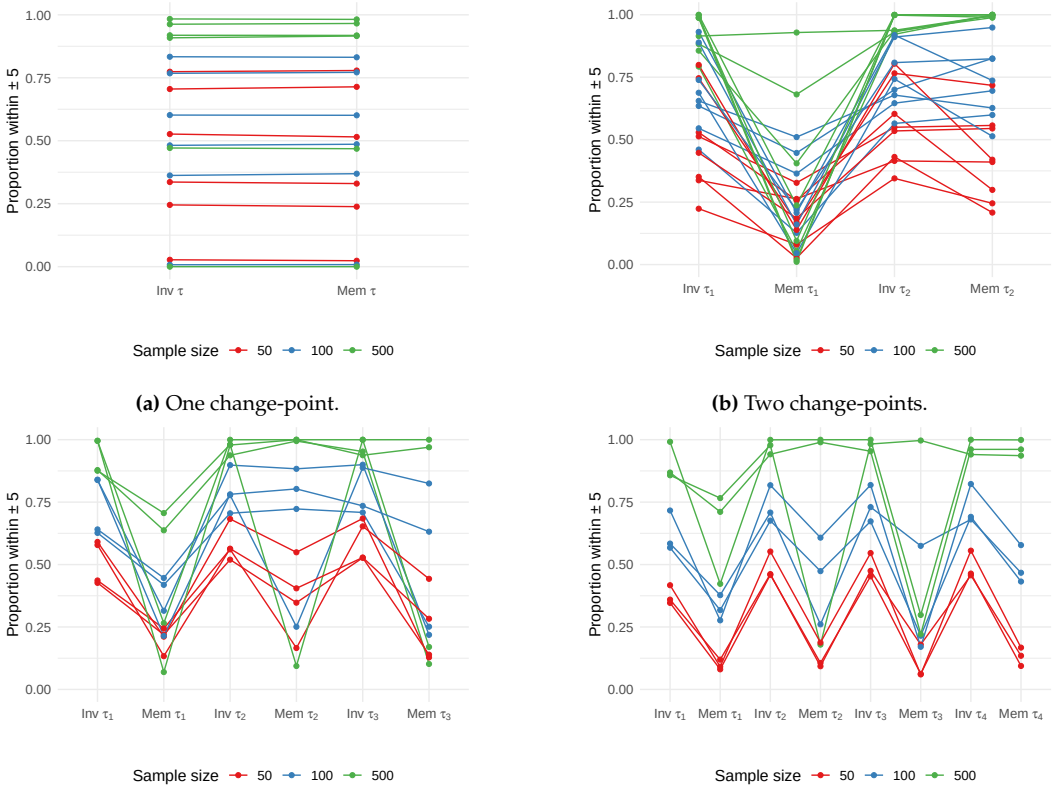
(c) Three change-points.

(d) Four change-points.

Figure 1: Accuracy of change-point simulations for the exponential distribution. The y-axis represents the proportion of change-point estimates ($\hat{\tau}$) within 10% of the known value. "Inv" refers to the inverse hazard method and "Mem" refers to the memoryless method. This figure demonstrates that accuracy is higher with a larger sample size and is similar for both simulation methods.

those scenarios, both simulation methods severely underestimated τ when the true value was 80. When simulating two change-points (Figure 2b), accuracy of the first change-point was often much lower when using the memoryless method, especially with a larger sample size. All except one of the simulation scenarios where the known value of τ_1 did not fall within the 95% confidence interval of the average estimate for the memoryless method had a sample size of 500. Accuracy was lower for all change-points in the three change-point simulations when using the memoryless method and the discrepancies between the two methods were larger for larger sample sizes (Figure 2c). In almost half of the simulation scenarios at least one value of τ_i was not within the 95% confidence interval of the mean estimate for the memoryless method and all except one of those scenarios had a sample size of 500. When simulating four change-points (Figure 2d), for most scenarios there was a large drop in accuracy at the first and third change-point for the memoryless method compared to the inverse hazard method. At the second and fourth change-points, however, there was a drop in accuracy for sample sizes of 50 and 100 whereas most simulation scenarios with a sample size of 500 had similar accuracy between the two methods. Every simulation scenario for four change-points with a sample size of 500 using the memoryless method had at least one change-point where the 95% confidence interval did not include the known value of τ_i. The known value of τ_4 was, however, included in the 95% confidence interval for every scenario with a sample size of 500.

We suspected that the inaccurate estimates using the Weibull distribution were due to inaccuracies in estimating the shape parameter γ, which is assumed constant across all time intervals. Indeed, γ was often under-estimated as seen in Figure 3. Values of γ, however, were similar for both methods except when there were three change-points (Figure 3c), in which case the estimates of γ using the inverse hazard method were closer to the known value of two. We were unable to estimate γ for the simulations using the memoryless method when there was a sample size of 50 (Figure 3d).



(a) One change-point.

(b) Two change-points.

(c) Three change-points.

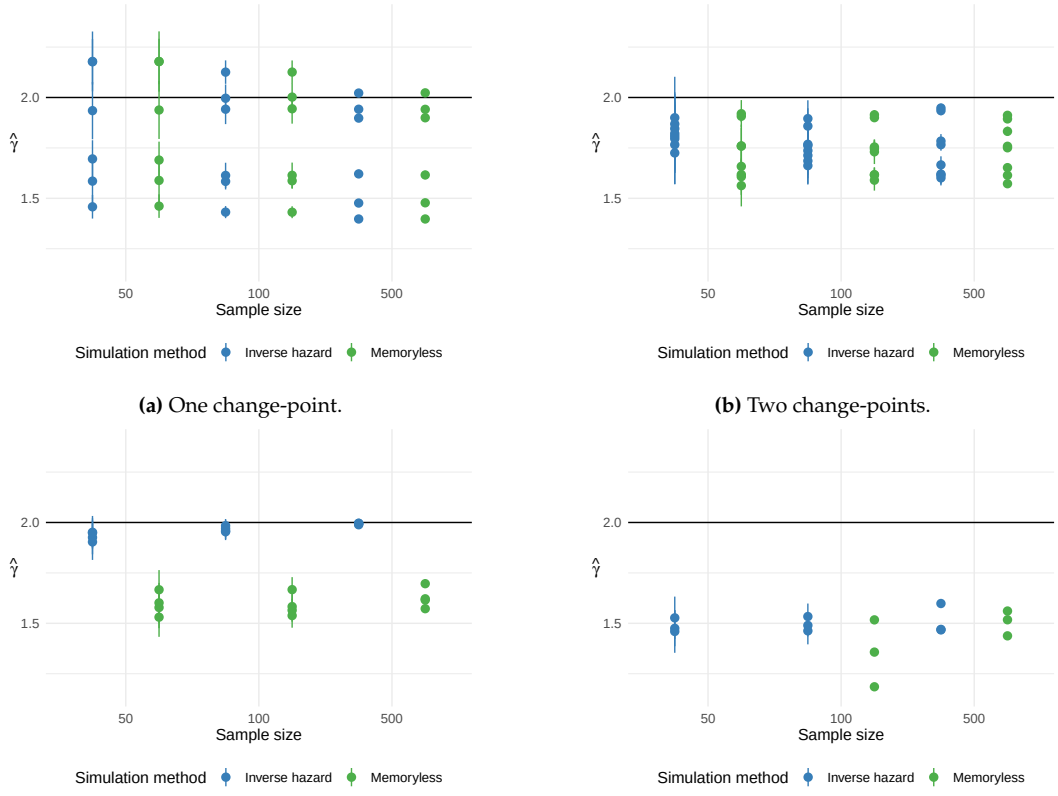
(d) Four change-points.

Figure 2: Accuracy of change-point simulations for the Weibull distribution. The y-axis represents the proportion of change-point estimates ($\hat{\tau}$) within 10% of the known value. "Inv" refers to the inverse hazard method and "Mem" refers to the memoryless method. This figure demonstrates that accuracy is generally higher with larger sample sizes and when using the inverse hazard method.

5 Summary

The R package **cpsurvsim** provides implementation of the standard method of simulating from a distribution, using the inverse CDF, and a new method that exploits the memoryless property of survival analysis. When simulating from the exponential distribution with multiple change-points, these methods have comparable performance. Simulating multiple change-points from the Weibull hazard, however, suggested that the inverse hazard method produces more accurate estimates of the change-points τ_i. The accuracy of the exponential simulations suffered when the sample size was less than 500 whereas in some cases, simulations of the Weibull distribution had worse accuracy with a sample size of 500. In practice, change-point hazard models are often applied to data from large longitudinal cohort studies where the sample size is very large (e.g., Goodman et al. (2011) and Williams and Kim (2013)). These results suggest that larger sample sizes are preferred when using an exponential model but to use caution even with a large sample when using the Weibull model. We hope that having an R package for simulating data from multiple change-point hazard distributions will aid in the development of extensions and alternatives to our research on tests for multiple change-points (Hochheimer and Sabo, 2021).

The inspiration to develop the memoryless simulation method and test it came from observing the shortcomings of the inverse hazard method in our research. The memoryless method performs better in some simulation scenarios, which led us to implement both methods in this R package. This simulation study, however, suggests that in the majority of cases the inverse hazard method simulates values of τ_i more accurately. Our simulation study also highlighted accuracy issues with both methods when simulating data from sample sizes of 50 or 100, which we suspect are due to using a relatively small amount of data to estimate several model parameters. One should consider exploring other methods to simulate a multiple change-point distribution with a small sample size. The acceptance-rejection method, for example, may produce more accurate parameter estimates at the cost of more computational time needed to reach the desired sample size, a cost that might be worthwhile if the sample size is smaller to begin with (Rizzo, 2007). Alternatively, one might run a simulation study to determine which of these two methods is best suited for their specific parameters.



(a) One change-point.

(b) Two change-points.

(c) Three change-points.

(d) Four change-points.

Figure 3: Estimated values of shape parameter γ for the Weibull distribution. Dots indicate the average estimated value for each simulation scenario with the vertical lines representing the 95% CI. The solid horizontal line represents the known value of γ. This plot demonstrates inaccurate estimates of γ for both simulation methods except when there are three change-points, in which case the estimates are more accurate using the inverse hazard method.

An important limitation of the `exp_cdfsim` and `weib_cdfsim` functions is that they only accommodate up to four change-points. While it's possible to have more than this many change-points in a dataset, it's also important to make sure that there is a meaningful interpretation for multiple change-points. Also, **cpsurvsim** only accommodates type I right censoring. For the Weibull distribution, γ is assumed fixed for every interval between change-points. In our simulation study, we only estimated an overall value of γ due to convergence issues when trying to estimate it within each interval between change-points. In an effort to be concise, the accuracy of the scale parameters θ_i are not discussed here, however, in some cases this parameter may be of more interest than the change-point τ. Thus, we briefly discuss these results in the appendix. Future versions of **cpsurvsim** could incorporate additional features such as accommodating informative censoring.

6 Acknowledgments

Thank you to Dr. Sarah Ratcliffe for her guidance and for assisting with running these simulations.

Bibliography

A. Allignol and A. Latouche. *Task View: Survival Analysis*, 2020. URL https://www.r-pkg.org/ctv/Survival. METACRAN. [p196]

A. Araújo, L. Meira-Machado, and S. Faria. *genSurv: Generating Multi-State Survival Data*, 2015. URL http://CRAN.R-project.org/package=genSurv. R package version 1.0.3. [p197]

N. Blaser, L. Salazar Vizcaya, J. Estill, C. Zahnd, B. Kalesan, M. Egger, O. Keiser, and T. Gsponer. gems: An r package for simulating from disease progression models. *Journal of Statistical Software*, 64(10): 1–22, 2015. doi: 10.18637/jss.v064.i10. [p197]

S. Brilleman. *simsurv: Simulate Survival Data*, 2019. URL https://CRAN.R-project.org/package=simsurv. R package version 0.2.3. [p197]

X. Cai, Y. Tian, and W. Ning. Modified information approach for detecting change points in piecewise linear failure rate function. *Statistics & Probability Letters*, 125:130–140, 2017. ISSN 0167-7152. doi: 10.1016/j.spl.2017.02.005. [p196]

A. G. Chapple. *SimSCRPiecewise: Simulates Univariate and Semi-Competing Risks Data Given Covariates and Piecewise Exponential Baseline Hazards'*, 2016. URL https://CRAN.R-project.org/package=SimSCRPiecewise. R package version 0.1.1. [p196, 197]

J.-F. Dupuy. Estimation in a change-point hazard regression model. *Statistics & Probability letters*, 76(2): 182–190, 2006. ISSN 0167-7152. doi: 10.1016/j.spl.2005.07.013. [p196]

C. Gandrud. simph: An r package for illustrating estimates from cox proportional hazard models including for interactive and nonlinear effects. *Journal of Statistical Software*, 65(3):1–20, 2015. doi: 10.18637/jss.v065.i03. [p197]

T. A. Gerds. *prodlim: Product-Limit Estimation for Censored Event History Analysis*, 2018. URL https://CRAN.R-project.org/package=prodlim. R package version 2018.04.18. [p197]

I. Gijbels and U. Gurler. Estimation of a change point in a hazard function based on censored data. *Lifetime Data Analysis*, 9(4):395–411, 2003. ISSN 1380-7870. doi: 10.1023/B:LIDA.0000012424.71723.9d. [p196]

M. S. Goodman, Y. Li, and R. C. Tiwari. Detecting multiple change points in piecewise constant hazard functions. *Journal of Applied Statistics*, 38(11):2523–2532, 2011. ISSN 0266-4763. doi: 10.1080/02664763.2011.559209. [p196, 202]

G. Han, M. J. Schell, and J. Kim. Improved survival modeling in cancer research using a reduced piecewise exponential approach. *Statistics in Medicine*, 33(1):59–73, 2014. doi: 10.1002/sim.5915. [p196]

P. He, L. Fang, and Z. Su. A sequential testing approach to detecting multiple change points in the proportional hazards model. *Statistics in Medicine*, 32(7):1239–1245, 2013. doi: 10.1002/sim.5605. [p196]

D. J. Hendry. Data generation for the cox proportional hazards model with time-dependent covariates: A method for medical researchers. *Statistics in Medicine*, 33(3):436–454, 2014. doi: 10.1002/sim.5945. [p196]

C. Hochheimer. *cpsurvsim: Simulating Survival Data from Change-Point Hazard Distributions*, 2021. URL http://github.com/camillejo/cpsurvsim. [p196, 197]

C. J. Hochheimer and R. T. Sabo. Testing for phases of dropout attrition using change-point hazard models. *Journal of Survey Statistics and Methodology*, 09 2021. ISSN 2325-0984. doi: 10.1093/jssam/smab030. URL https://doi.org/10.1093/jssam/smab030. [p202]

J. Kropko and J. J. Harden. *coxed: Duration-Based Quantities of Interest for the Cox Proportional Hazards Model*, 2019. URL https://CRAN.R-project.org/package=coxed. R package version 0.2.4. [p197]

S. Krügel, A. R. Brazzale, and H. Kuechenhoff. *CPsurv: Nonparametric Change Point Estimation for Survival Data*, 2017. URL https://CRAN.R-project.org/package=CPsurv. R package version 1.0.0. [p196, 197]

M. Liu, W. Lu, and Y. Shao. A monte carlo approach for change-point detection in the cox proportional hazards model. *Statistics in medicine*, 27(19):3894–3909, 2008. ISSN 1097-0258. doi: 10.1002/sim.3214. [p196]

L. Meira-Machado and S. Faria. A simulation study comparing modeling approaches in an illness-death multi-state model. *Communications in Statistics - Simulation and Computation*, 43(5):929–946, 2014. ISSN 1532-4141. doi: 10.1080/03610918.2012.718841. [p197]

M. E. Montez-Rath, K. Kapphahn, M. B. Mathur, A. A. Mitani, D. J. Hendry, and M. Desai. Guidelines for generating right-censored outcomes from a cox model extended to accommodate time-varying covariates. *Journal of Modern Applied Statistical Methods*, 16(1):6, 2017a. doi: 10.22237/jmasm/1493597100. [p196]

M. E. Montez-Rath, K. Kapphahn, M. B. Mathur, N. Purington, V. R. Joyce, and M. Desai. Simulating realistically complex comparative effectiveness studies with time-varying covariates and right-censored outcomes. *arXiv*, 09 2017b. doi: https://arxiv.org/abs/1709.10074. [p197]

D. Moriña and A. Navarro. The r package survsim for the simulation of simple and complex survival data. *Journal of Statistical Software*, 59(2):1–20, 2014. doi: 10.18637/jss.v059.i02. [p197]

A. Peters and T. Hothorn. *ipred: Improved Predictors*, 2019. URL https://CRAN.R-project.org/package=ipred. R package version 0.9-9. [p197]

L. Qian and W. Zhang. *Multiple Change-Point Detection in Piecewise Exponential Hazard Regression Models with Long-Term Survivors and Right Censoring*, book section 18. Springer International Publishing, Switzerland, 2014. doi: 10.1007/978-3-319-02651-0_18. [p196]

H. Reulen. *simMSM: Simulation of Event Histories for Multi-State Models*, 2015. URL https://CRAN.R-project.org/package=simMSM. R package version 1.1.41. [p197]

M. L. Rizzo. *Statistical Computing with R*. CRC Press, 2007. ISBN 9781498786591. [p202]

O. Rojas, F. Bulmaro, and J. Hernández. Modelo de riesgo con un punto de cambio y covariables dependientes del tiempo. *Revista Investigación Operacional*, 32:114–122, 2011. [p196]

M.-P. Sylvestre, T. Evans, T. MacKenzie, and M. Abrahamowicz. *PermAlgo: Permutational Algorithm to Generate Event Times Conditional on a Covariate Matrix Including Time-Dependent Covariates*, 2010. URL https://cran.r-project.org/package=PermAlgo. R package version 1.1. [p197]

R. Walke. Example for a piecewise constant hazard data simulation in r. Report, Max Planck Institute for Demographic Research, 2010. URL https://www.demogr.mpg.de/papers/technicalreports/tr-2010-003.pdf. [p196, 198]

T. Welchowski and M. Schmid. *discSurv: Discrete Time Survival Analysis*, 2019. URL https://CRAN.R-project.org/package=discSurv. R package version 1.4.0. [p197]

M. R. Williams and D. Y. Kim. A test for an abrupt change in weibull hazard functions with staggered entry and type i censoring. *Communications in Statistics - Theory and Methods*, 42(11):1922–1933, 2013. ISSN 0361-0926. doi: 10.1080/03610926.2011.600505. [p202]

G. Y. C. Wong, Q. Diao, and Q. Yu. Piecewise proportional hazards models with interval-censored data. *Journal of Statistical Computation and Simulation*, 88(1):140–155, 2018. doi: 10.1080/00949655.2017.1380645. [p196]

C. Q. Wu, L. C. Zhao, and Y. H. Wu. Estimation in change-point hazard function models. *Statistics & Probability Letters*, 63(1):41–48, 2003. ISSN 0167-7152. doi: 10.1016/S0167-7152(03)00047-6. [p196]

Y. C. Yao. Maximum-likelihood-estimation in hazard rate models with a change-point. *Communications in Statistics - Theory and Methods*, 15(8):2455–2466, 1986. ISSN 0361-0926. doi: 10.1080/03610928608829261. [p196]

M. Zhou. Understanding the cox regression models with time-change covariates. *The American Statistician*, 55(2):153–155, 2001. doi: 10.1198/000313001750358491. [p196, 199]

S. Zinn. *MicSim: Performing Continuous-Time Microsimulation*, 2018. URL https://CRAN.R-project.org/package=MicSim. R package version 1.0.13. [p197]

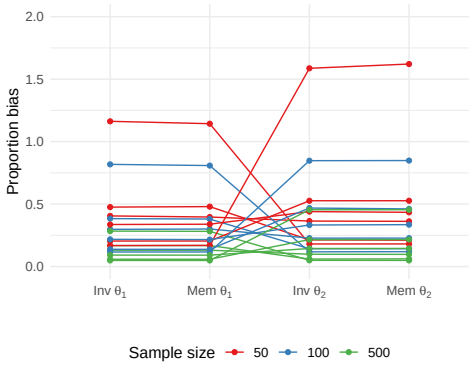
Camille J. Hochheimer, PhD
Department of Biostatistics and Informatics
Colorado School of Public Health
13001 East 17th Place, 4th Floor West, Aurora, CO 80045
USA
ORCiD: 0000-0002-0984-0909
camille.hochheimer@cuanschutz.edu

Roy T. Sabo, PhD
Department of Biostatistics
Virginia Commonwealth University
PO Box 980032, Richmond, VA 23298-0032
USA
ORCiD: 0000-0001-9159-4876
roy.sabo@vcuhealth.org

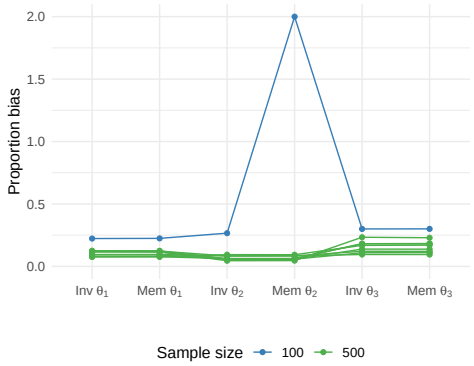
1 Appendix: Analysis of scale parameters

While the change-points can be estimated without knowing the values of the scale parameters, the reverse is not possible. Thus, we used the estimated values of the change-points in order to estimate values of θ_i. As the number of change-points increased, so did the difficulty in estimating values of θ_i, especially with a smaller sample size.

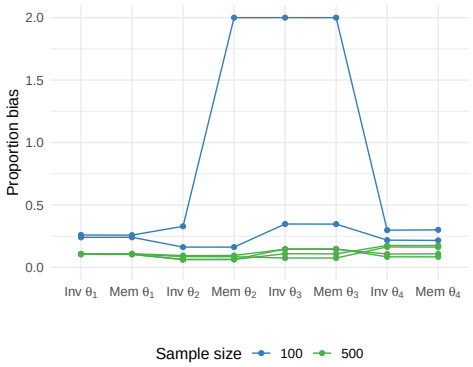
With a few exceptions, the estimates of θ_i for the exponential distribution were similar between both methods (Figure 4). These exceptions were θ_2 in the two change-point (Figure 4b) and three change-point models (Figure 4c), where the memoryless method with a sample size of 100 had a much larger proportion of bias. We were only able to estimate the shape parameters for the four change-point model when the sample size was 500.



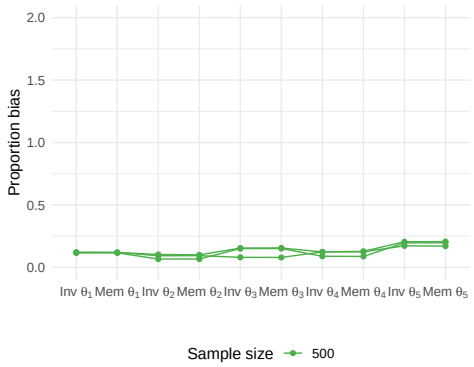
(a) One change-point.



(b) Two change-points.



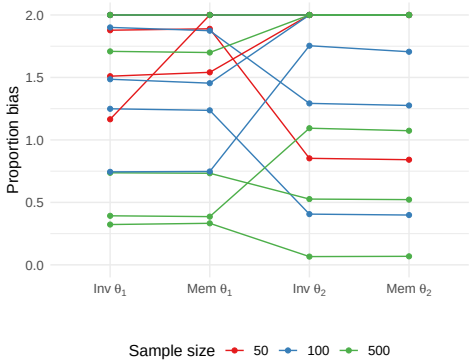
(c) Three change-points.



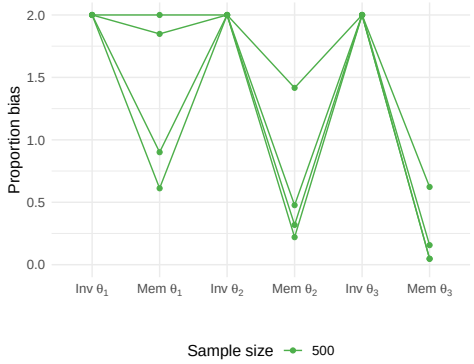
(d) Four change-points.

Figure 4: Accuracy of scale parameter $\hat{\theta}_i$ for the exponential distribution. The y-axis represents the average proportion of bias of $\hat{\theta}_i$ relative to the known value of θ_i. A proportion of bias of 2 represents estimates with at least 200% bias. "Inv" refers to the inverse hazard method and "Mem" refers to the memoryless method. This figure demonstrates that bias was generally similar between simulation methods with a few exceptions where bias was larger using the memoryless method.

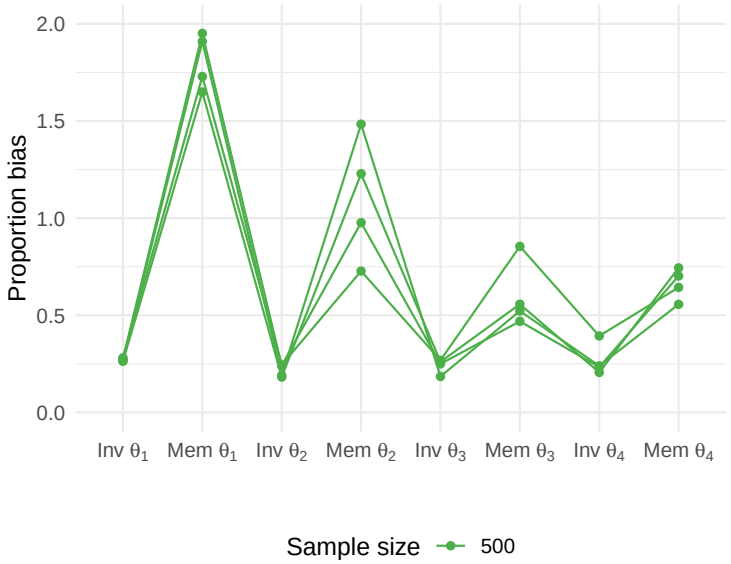
Estimates of θ_i for the one change-point Weibull model were similar across simulation methods but bias was high even when the sample size was large (Figure 5a). Bias was generally smaller when using the memoryless method to estimate θ_i in the two change-point Weibull model (Figure 5b). On the other hand, bias was larger when using the memoryless method to estimate the shape parameter for the three change-point Weibull model (Figure 5c). We were unable to estimate θ using the results from the memoryless method for any of the four change-point simulations.



(a) One change-point.



(b) Two change-points.



(c) Three change-points.

Figure 5: Accuracy of scale parameter $\hat{\theta}_i$ for the Weibull distribution. The y-axis represents the average proportion of bias of $\hat{\theta}_i$ relative to the known value of θ_i. A proportion of bias of 2 represents estimates with at least 200% bias. "Inv" refers to the inverse hazard method and "Mem" refers to the memoryless method. This figure demonstrates similar bias between methods when there is one change-point, smaller bias using the memoryless method when there are two change-points, and smaller bias using the inverse hazard method when there are three change-points.

starvars: An R Package for Analysing Nonlinearities in Multivariate Time Series

by Andrea Bucci, Giulio Palomba and Eduardo Rossi

Abstract Although linear autoregressive models are useful to practitioners in different fields, often a nonlinear specification would be more appropriate in time series analysis. In general, there are many alternative approaches to nonlinearity modelling, one consists in assuming multiple regimes. Among the possible specifications that account for regime changes in the multivariate framework, smooth transition models are the most general, since they nest both linear and threshold autoregressive models. This paper introduces the **starvars** package which estimates and predicts the Vector Logistic Smooth Transition model in a very general setting which also includes predetermined variables. In comparison to the existing R packages, **starvars** offers the estimation of the Vector Smooth Transition model both by maximum likelihood and nonlinear least squares. The package allows also to test for nonlinearity in a multivariate setting and detect the presence of common breaks. Furthermore, the package computes multi-step-ahead forecasts. Finally, an illustration with financial time series is provided to show its usage.

1 Introduction

Many economic and financial time series often behave differently during stress periods for the economic activity. For example, during the subprime mortgage financial crisis, the relationship between the financial sector and macroeconomic quantities changed justifying the use of a nonlinear model. The same is also true in the analysis of monetary policy, where positive and negative monetary policy shocks may have asymmetric effects, or in the investigation of the effectiveness of a fiscal policy, where some fiscal policy measures may depend on the phase of the business cycle, see for example Caggiano et al. (2015). When asymmetric effects are observed, the time series may follow different regimes. In order to understand the dynamics of such processes, Quandt (1958, 1960) firstly proposed a model where the coefficients of a linear model change in relation to the value of an observable stochastic variable. Afterwards, these models have been extended to time series analysis. Tong (1978) and Teräsvirta and Lim (1980) introduced the threshold autoregressive model, while Teräsvirta (1994) imagined that the transition between regimes could be smooth, which leads to the smooth transition autoregressive model (STAR) for univariate time series.

Since researchers are often interested in understanding the dynamics of time series in a multivariate framework, regime-switching models have also been extended to include multiple dependent variables. A vector nonlinear model was introduced by Tsay (1998), who defined a Threshold Vector Autoregressive (TVAR) model with a single threshold variable controlling the switching mechanism in each equation. The first vector model with a smooth transition was the smooth transition vector error-correction model (STVECM) introduced by Rothman, van Dijk, and Franses (2001). In this model, the same transition function controls the transition in each equation. Camacho (2004) proposed a bivariate logistic smooth transition model with the possibility to include exogenous regressors and specify a different transition variable for each equation. For a recent survey of vector TAR and STAR models, see Hubrich and Teräsvirta (2013). More recently, Teräsvirta and Yang (2014a) presented a modelling strategy for building a Vector Logistic Smooth Transition Regression (VLSTAR). This strategy includes linearity and misspecification tests for the conditional mean, and testing the constancy of the error covariance matrix.

This article summarizes the procedure proposed in Teräsvirta and Yang (2014a) and illustrates the **starvars** package in R for estimating and testing of the VLSTAR model with a single transition variable. Several packages for the estimation of the univariate logistic autoregressive model (LSTAR) are already present in R. For example, Di Narzo, Aznarte, Stigler, and Tsung-wu (2020) in their **tsDyn** package provide functions to estimate and forecast both the STAR and the LSTAR models. Unfortunately, the **tsDyn** package, which focuses on nonlinear models in general, only allows for the estimation of a multivariate Threshold Vector Autoregressive (TVAR) model and does not allow for the inclusion of exogenous regressors. The **RSTAR** package, implemented by Balcilar (2016), estimates, forecasts, and analyses the smooth transition autoregressive model in the univariate case. Another possible way to model regime switches in a multivariate framework is through the **MSBVAR** by Brandt (2016), capable of estimating a Markov-switching autoregressive model. Still, this package does not permit to evaluate the relationship between the dependent variables and possible explanatory variables.

The here presented R package **starvars** (Bucci et al., 2022) is conceived for the nonlinear specification with a VLSTAR model of the relationship of multivariate time series exhibiting smooth nonlinear

relationships with both their lags and a set of explanatory variables. Even though this model has been mainly applied in financial setups, it could be used in all fields in which the nature of the dynamics of the dependent variables could be conceived somehow nonlinear and, specifically, following a logistic smooth transition model. The functionalities of the **starvars** package include: (i) modelling strategy, such as joint linearity testing of multivariate time series, or detecting the presence of co-breaks, (ii) estimation and (iii) prediction of the VLSTAR model, (iv) construction of realized covariances from high and low-frequency financial prices or returns. Two datasets (`Realized` and `techprices`) are included in the R package **starvars**. The former entails monthly observations for realized co-volatilities between the S&P 500, the Nikkei, the FTSE and the DAX indexes, the growth rate of the dividend yield and the earning price ratio, and the first difference of the inflation rate in the U.S., United Kingdom, Japan and Germany. The latter includes the data used in the example with the daily closing stock prices of Google, Microsoft and Amazon.

The outline of the paper is as follows. The following sections review the specification of the VLSTAR model, referring to Teräsvirta and Yang (2014a), and illustrate how to estimate and make predictions through the **starvars** package. We then present an empirical application to stock price data, while the last section concludes.

2 The Vector Logistic Smooth Transition Autoregressive Model

Assuming an $n \times 1$ vector of dependent time series, y_t, the multivariate smooth transition model introduced by Teräsvirta and Yang (2014a) can be written as follows

$$\begin{aligned} y_t &= \mu_0 + \sum_{j=1}^{p} \Phi_{0,j}\, y_{t-j} + A_0 x_t + G_t\left(s_t;\gamma,c\right)\left[\mu_1 + \sum_{j=1}^{p} \Phi_{1,j}\, y_{t-j} + A_1 x_t\right] + \varepsilon_t \\ &= \mu_0 + G_t\left(s_t;\gamma,c\right)\mu_1 + \sum_{j=1}^{p}\left[\Phi_{0,j} + G_t\left(s_t;\gamma,c\right)\Phi_{1,j}\right] y_{t-j} + \left[A_0 + G_t\left(s_t;\gamma,c\right)A_1\right] x_t + \varepsilon_t, \quad (1) \end{aligned}$$

where μ_0 and μ_1 are the $n \times 1$ vectors of intercepts, $\Phi_{0,j}$ and $\Phi_{1,j}$ are square $n \times n$ matrices of parameters with lags $j = 1, 2, \ldots, p$, A_0 and A_1 are $n \times k$ matrices of parameters, x_t is the $k \times 1$ vector of exogenous variables and ε_t is the innovation. $G_t\left(s_t;\gamma,c\right)$ is a $n \times n$ diagonal matrix of transition function at time t, such that

$$G_t\left(s_t;\gamma,c\right) = \operatorname{diag}\left\{G_{1,t}\left(s_{1,t};\gamma_1,c_1\right), G_{2,t}\left(s_{2,t};\gamma_2,c_2\right), \ldots, G_{n,t}\left(s_{n,t};\gamma_n,c_n\right)\right\}, \quad (2)$$

where γ_i and c_i are the scale and the threshold parameters for the i-th equation, for $i = 1, \ldots, n$.

In the VLSTAR model, each element of G_t is specified as a logistic function

$$G_{i,t}\left(s_{i,t};\gamma_i,c_i\right) = \left[1 + \exp\left\{-\gamma_i\left(s_{i,t} - c_i\right)\right\}\right]^{-1}. \quad (3)$$

Let $B = \left[G_t^{-1}\mu_0 + \mu_1 \quad G_t^{-1}\Phi_{0,1} + \Phi_{1,1} \quad G_t^{-1}\Phi_{0,2} + \Phi_{1,2} \quad \ldots \quad G_t^{-1}\Phi_{0,p} + \Phi_{1,p} \quad G_t^{-1}A_0 + A_1\right]'$, by reformulating Equation (1) as in Teräsvirta and Yang (2014a) and extending for the presence of m regimes, Equation (1) becomes

$$y_t = \left\{\sum_{r=1}^{m} G_t^{r-1} B_r'\right\} z_t + \varepsilon_t = \begin{bmatrix} I_n & G_t^1 & \ldots & G_t^{m-1} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_m \end{bmatrix} z_t + \varepsilon_t = \tilde{G}_t \tilde{B}'\, z_t + \varepsilon_t, \quad (4)$$

where $\tilde{G}_t$ is a matrix of dimension $n \times mn$, $z_t = \begin{bmatrix} 1 & y_{t-1}' & y_{t-2}' & \ldots & y_{t-p}' & x_t' \end{bmatrix}'$, $\tilde{B}$ is a $(1 + k + pn) \times mn$ matrix and $G_t^0 = I_n$ is an identity matrix indicating that no transitions are allowed before the first change of regime. This equation defines the VLSTAR model with m regimes and p lags for the dependent variables.

The logistic function in Equation (3) is accordingly modified as follows

$$G_{i,t}^r\left(s_{i,t}^r;\gamma_i^r,c_i^r\right) = \left[1 + \exp\left\{-\gamma_i^r(s_{i,t}^r - c_i^r)\right\}\right]^{-1}, \quad (5)$$

for $i = 1, 2, \ldots, n$ and $r = 0, 1, \ldots, m-1$.

Joint linearity test

The VLSTAR specification procedure follows several steps. Firstly, the researcher should test whether the relationship between y_t and z_t can be linear. This is crucial, since several nonlinear models, like smooth transition and switching regression models, are not identified when the data-generating process is linear. With multivariate dependent variables, linearity can be tested equation by equation, using the Lagrange Multiplier (LM) test, as proposed by Luukkonen, Saikkonen, and Teräsvirta (1988), Teräsvirta (1994) and Teräsvirta, Tjøstheim, and Granger (2010), or it may be tested simultaneously, as introduced by Hubrich and Teräsvirta (2013) and Teräsvirta and Yang (2014b).

The LM type statistic can be computed, as further suggested by Teräsvirta and Yang (2014b), using a multi-step procedure:

1. estimation of the linear model, *i.e.* the restricted VLSTAR with $\gamma = 0$;
2. save a collection of the residuals $(\tilde{\varepsilon}_t)$ from step 1 to create the residual matrix $\tilde{E}$ of dimension $T \times n$;
3. computation of the residual sum of squares matrix, $Q = \tilde{E}'\tilde{E}$;
4. regression of $\tilde{E}$ on X and $V = (v_1', \ldots, v_T')'$, where $v_t = \left(z_t' s_t, z_t' s_t^2, \ldots, z_t' s_t^d\right)$ and s_t^d is the d-th order Taylor expansion of the logistic function (in our package $d = 3$, *i.e.* a third-order Taylor expansion has been used);
5. creation of the residual matrix, $\tilde{\Xi}$, from step 4 and the residual sum of square matrix, $\tilde{\Xi}'\tilde{\Xi}$;
6. computation of the test statistic

$$LM = T\left\{Q^{-1}Q - \tilde{\Xi}'\tilde{\Xi}\right\} = T\left(p - tr\left\{Q^{-1}\tilde{\Xi}'\tilde{\Xi}\right\}\right) \sim \chi^2_{dn(np+1)}. \tag{6}$$

   where $tr\{\cdot\}$ is the trace of the matrix.

In the R package **starvars**, the joint linearity test can be performed by using the function `VLSTARjoint`, which takes the following arguments.

- `y`: a `data.frame` or `matrix` containing the T observations for the n time series whose linearity should be tested;
- `exo`: an optional argument containing a `data.frame` or `matrix` of k explanatory variables;
- `st`: a vector with the observations of the single transition variable (s_t), or a matrix with a set of potential transition variables;
- `st.choice`: when the choice of the transition variable among a set of candidates should be based on the linearity test, this argument should be set equal to `TRUE`. In such a case, the variable in the matrix `st` which results in a higher LM statistics is the one chosen as the transition variable;
- `alpha`: a decimal value comprised between 0 and 1 ($\alpha \in [0, 1]$) representing the confidence level, set to 0.05 by default.

In this case, the residuals $\tilde{\varepsilon}_t$ used in step 2 of the above-mentioned procedure are obtained through a VAR(p) estimation of the restricted model in step 1. This is done through the `VAR` function from R package **vars**, with an automatically selected number of lags, p.

```
VLSTARjoint(y, exo, st, st.choice = FALSE, alpha = 0.05)
```

The function `VLSTARjoint` returns a list object with a class attribute `"VLSTARjoint"`, for which `print` method exists, with three elements: the value(s) of the Lagrange Multiplier value (LM), the p-value(s) of the test and the critical value.

Furthermore, the specification of the VLSTAR model foresees the definition of the number of regimes to be used in the model (see Appendix A for further details). The function `multiCUMSUM` allows determining the number of common breaks and where they are located.

```
multiCUMSUM(data, conf.level = 0.95, max.breaks = 7)
```

The arguments necessary to detect the common breaks are: a matrix of $T \times n$ of time series, in the argument `data`; the confidence level in `conf.level`, set by default at 0.95; the number of maximum common breaks (between 1 and 7) to be identified, through `max.breaks`. The output is returned in a list with a class attribute `"multiCUMSUM"`, which can be passed through the `print` function. The first element of the returned list object is a matrix with the test statistics Λ_T and Ω_T (see Equation (18) in Appendix A for details). The list further reports the index of the common breaks detected and the correspondent dates, as long as the critical values for both Λ_T and Ω_T.

VLSTAR Estimation

As widely discussed in Teräsvirta and Yang (2014a), a VLSTAR model can be estimated through a nonlinear Least Square (NLS) or a maximum likelihood (ML) model.

In both cases, the optimization algorithm may converge to some local minima, attributing to the definition of valid starting values of the estimated parameters a special relevance. If there is no clear indication of the initial values of γ and c, this can be done by implementing a grid search. Thus, a discrete grid in the parameter space is created to obtain the estimates of B conditionally on each point in the grid. The initial pairs of γ and c producing the smallest sum of squared residuals are chosen as initial values. A pair of these parameters for each equation is selected unless common parameters are assumed. Given their values, the model is linear in parameters.

The searching grid algorithm works as follows:

1. construction of the grid for γ and c, computing the vector of parameters for each point in the grid;
2. estimation of $\tilde{B}$ in each equation through NLS and computation of the residual sum of squares, Q;
3. find the pairs of γ and c providing the smallest Q which will be the starting γ_0 and c_0;
4. estimation of parameters, $\tilde{B}$, via NLS or ML;
5. estimation of γ and c for each equation given the parameters found in step 4;
6. repeat steps 4 and 5 until convergence.

The **starvars** package allows the user to implement a searching grid algorithm to obtain the initial values of c and γ. Specifically, the practitioner may obtain initial values through the `startingVLSTAR` function among a set of potential values. For example, by providing `n.combi`$= 50$, 50 values of γ and c are combined in a grid of 2500 couples of values as in step 1 of the former procedure. The values of the grid for γ range from 0 to 100, while the values of c range from minimum to maximum of each dependent variable.

The `startingVLSTAR` function requires several arguments. A `data.frame` or a `matrix` of dimension $T \times n$ containing the dependent variables of the model, representing `y`. An optional argument, `exo`, contains possible explanatory variables and can be specified as a `data.frame` or a `matrix` with the same length of `y` and k columns. The lag-order p should be specified as an integer. The number of regimes in the model is set by the argument `m`, while the transition variable s_t of length T is specified in the argument `st`. The number of cores used to make parallel computation is specified through the `ncores` argument, while the argument `singlecgamma` works as follows:

- `singlecgamma = TRUE`: it is assumed a common pair of initial values for the entire model;
- `singlecgamma = FALSE`: a pair of c and γ is obtained for each of the equations.

```
startingVLSTAR(y, exo = NULL, p = 1,
m = 2, st = NULL, constant = TRUE,
n.combi = NULL, ncores = 2,
singlecgamma = FALSE)
```

VLSTAR Estimation via NLS

The NLS estimator is defined as the solution to the following optimisation problem

$$\hat{\theta}_{NLS} = \arg\min_{\theta} \sum_{t=1}^{T} \left(y_t - \tilde{G}_t \tilde{B}' z_t\right)' \left(y_t - \tilde{G}_t \tilde{B}' z_t\right) \tag{7}$$

where θ is the set of parameters to be estimated.

In the aforementioned algorithm, the vectorization of the NLS estimates of $\tilde{B}$ for step 4, given the values of γ and c, is equal to:

$$\text{vec}(\tilde{B})_{NLS} = \left[T^{-1} \sum_{t=1}^{T} \left(\tilde{G}_t \tilde{G}_t'\right) \otimes \left(z_t z_t'\right)\right]^{-1} \left[T^{-1} \sum_{t=1}^{T} \text{vec}\left(z_t y_t' \tilde{G}_t'\right)\right]. \tag{8}$$

The estimated errors covariance matrix is given by

$$\hat{\Omega}_{NLS} = T^{-1} \hat{E}' \hat{E}, \tag{9}$$

where $\hat{E} = (\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_n)'$ is a $T \times n$ matrix, and $\hat{\varepsilon}_t = y_t - \tilde{G}_t \tilde{B}'_{NLS} z_t$ is a column vector of residuals. This is used to obtain the first iterative ML estimation in the previous algorithm in step 4.

VLSTAR Estimation via ML

To estimate a VLSTAR model via ML, it must be assumed that $\varepsilon_t \sim i.i.d.N(0, \Omega)$. In this case, the model can be represented by the following multivariate conditional density function

$$f\left(y_t|\mathcal{I}_T;\theta\right) = (2\pi)^{-\frac{n}{2}} |\Omega|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(y_t - \tilde{G}_t\tilde{B}' z_t\right)' \Omega^{-1}\left(y_t - \tilde{G}_t\tilde{B}' z_t\right)\right\}, \tag{10}$$

where $\mathcal{I}_t$ is the information set at time t which contains all the exogenous variables x_t and all the lags of y_t.

In the first iteration of the algorithm presented in this section, Ω is estimated through Equation (9). Consequently, the ML estimator of θ is obtained by solving the optimization problem

$$\hat{\theta}_{ML} = arg\max_{\theta} \ell\left(y_t|\mathcal{I}_t;\theta\right). \tag{11}$$

Estimation in the starvars package

In the **starvars** package, the estimation of a VLSTAR model is handled with the function `VLSTAR`. By fitting such a model via this function, a list object with a class attribute `"VLSTAR"` is obtained. This function requires the same arguments of the `startingVLSTAR` function, except for the number of combinations. In addition, a `list` of `data.frame` or `matrix` containing starting values of c and γ, for each of the $m-1$ logistic functions as in Equation (5), must be passed through the argument `starting`. The user can choose the method used to estimate the coefficients among the 'ML' and the 'NLS' through the specification of the argument `method`. The argument `epsilon` is used as a convergence check while the argument `ncores` denotes the number of cores used in the parallel optimization of the objective function.

```
VLSTAR(y, exo = NULL, p = 1, m = 2, st = NULL, constant = TRUE,
starting = NULL,
method = c('ML', 'NLS'),
n.iter = 500, singlecgamma = TRUE,
epsilon = 10^(-3), ncores = NULL)
```

The `summary` method applied to an object derived from the `VLSTAR` function returns the sample size, along with the number of estimated parameters, the multivariate log-likelihood calculated as in Equation (10), and the estimated coefficients. We also provide other generic methods, such as `plot`, `AIC`, `BIC` and `logLok`. Similar to what is implemented in the R package **vars**, the `plot` function reports for each equation in the VLSTAR model the observed values of each time series, the fitted values and the residuals, as well as the autocorrelation and partial autocorrelation functions of the residuals. Since the logistic function plays a crucial role in VLSTAR models, the `plot` function shows also the plot of the logistic function for each dependent variable.

Forecasting a VLSTAR model

Time series prediction using nonlinear models has become widespread in the last few decades, even if the debate on the usefulness of such forecasts is still open (see Diebold and Nason, 1990; Kock and Teräsvirta, 2011). The forecasts of the nonlinear model, for more than one step ahead, can be generalised via numerical techniques. Given a nonlinear model

$$y_t = g\left(z_t, \theta\right) + \varepsilon_t, \tag{12}$$

where θ is a vector of parameters to be estimated, z_t is a combination of lagged values of y_t and exogenous variables x_t, and ε_t is a white noise with zero mean and constant variance σ^2, the forecast of y_{t+h} made at time t is equal to the conditional mean

$$\hat{y}_{t+h|t} = E\left\{y_{t+h}|\mathcal{I}_t\right\} = E\left\{g(z_{t+h-1})|\mathcal{I}_t\right\}. \tag{13}$$

where $\mathcal{I}_t$ is the information set at time t and ε_t is independent of $\mathcal{I}_{t-1}$.

When $h = 1$, the forecast $\hat{y}_{t+1} = g(z_t)$ is obtained from Equation (13); if $h \geq 2$, the prediction can only be calculated recursively using numerical techniques.

The nonlinearity in the VLSTAR model makes multi-period forecasting more complicated. In fact, forecasting two steps ahead is not straightforward, since we have

$$y_{t+2|t} = E\left(y_{t+2}|\mathcal{I}_t\right) = E\left\{\left[g(z_{t+2};\theta) + \varepsilon_{t+2}\right]|\mathcal{I}_t\right\} \tag{14}$$

and consequently

$$y_{t+2|t} = E\left\{\left[g(z_{t+2};\theta) + \varepsilon_{t+2}\right]|\mathcal{I}_t\right\} = \int_{-\infty}^{+\infty} g(z_{t+2}\theta)d\Phi(v)dv \tag{15}$$

where $\Phi(v)$ is the cumulative distribution function for ε_{t+1}. It follows that to obtain the $t+2$ forecast of y numerical integration would be necessary, while multiple integrations would be required for longer time horizons; see Lundbergh and Teräsvirta (2007).

The R package **starvars** can handle both one-step and multi-step-ahead forecasts of an object with a class attribute `"VLSTAR"`. One-step-ahead forecasts can be easily extended to the multivariate framework by modifying Equation (4) as follows

$$y_{t+1} = \tilde{G}_{t+1}\left(s_{t+1};\hat{\gamma},\hat{c}\right)\hat{\tilde{B}}'z_{t+1}$$

where $\hat{\tilde{B}}$ is the matrix of estimated parameters and $z_{t+1} = \left[1, y'_t, y'_{t-1}, \ldots, y'_{t-p+1}, x'_{t+1}\right]'$, while $\tilde{G}_{t+1}$ is calculated using estimated values of γ and c. Multi-step-ahead forecasts are slightly trickier to be found and several alternatives can be used. As shown in Lundbergh and Teräsvirta (2007) for the univariate case, multi-step-ahead forecasts can be obtained in three ways: naively, by Monte Carlo simulation and by bootstrapping. The method `predict` in the **starvars** package allows the user to choose between these methods through the argument `method`. When the `naive` method is chosen, the y_{t+h} forecasts are obtained as follows

$$y^{na}_{t+h} = \tilde{G}_{t+h}\left(s_{t+h};\hat{\gamma},\hat{c}\right)\hat{\tilde{B}}'z^{na}_{t+h}$$

where $z^{na}_{t+h} = \left[1, y'_{t+h-1}, \ldots, y'_{t+h-p}, x'_{t+h}\right]'$. If the transition variable is the lagged y_{t-s}, with $s < h$, the prediction of the i-th element of y is used as a new transition variable, otherwise the new value of s_t should be passed through the argument `st.new`. The index i is specified by the argument `st.num`, which denotes the column number of the dependent variable which should be used as a new transition variable. From Hubrich and Teräsvirta (2013), Kock and Teräsvirta (2011) and Teräsvirta et al. (2010), we know that these forecasts are biased. Thus, the practitioner may choose the `Monte Carlo` method. In this case, ε_{t+1} should be simulated using a properly defined error distribution. Let $\hat{B}_1 = \left[\hat{\mu}_0, \hat{\Phi}_{0,1}, \ldots \hat{\Phi}_{0,p}, \hat{A}_0\right]$ and $\hat{B}_2 = \left[\hat{\mu}_1, \hat{\Phi}_{1,1}, \ldots, \hat{\Phi}_{1,p}, \hat{A}_1\right]$, the multivariate version of the Monte Carlo method for h steps ahead is given by

$$y^{mc}_{t+h} = \hat{B}'_1 z_{t+h} + \frac{1}{M}\sum_{m=1}^{M} \tilde{G}_{t+h}\left(s_{t+h};\hat{\gamma},\hat{c}\right)\hat{B}'_2 z^{mc}_{t+h}$$

where $z^{mc}_{t+h} = \left[1, \left(y_{t+h-1} + \varepsilon^{mc}_{t+h}\right)', \ldots, \left(y_{t+h-p} + \varepsilon^{mc}_{t+h-p+1}\right)', x'_{t+h}\right]'$, ε^{mc}_{t+h} is a vector of errors sampled from a Multivariate Normal distribution with zero mean and covariance matrix $\hat{\Omega}$. In such a case, the interval forecasts are directly determined from the forecast density. Finally, the `bootstrap` method foresees that the multi-step-ahead forecasts are derived from

$$y^{bo}_{t+h} = \hat{B}'_1 z_{t+h} + \frac{1}{B}\sum_{b=1}^{B} \tilde{G}_{t+h}\left(s_{t+h};\hat{\gamma},\hat{c}\right)\hat{B}'_2 z^{bo}_{t+h}$$

where $z^{bo}_{t+h} = \left[1, \left(y_{t+h-1} + \varepsilon^{bo}_{t+h}\right)', \ldots, \left(y_{t+h-p} + \varepsilon^{bo}_{t+h-p+1}\right)', x'_{t+h}\right]'$, ε^{bo}_{t+h} is sampled from the $T \times n$ matrix of residuals. As in the case of the `Monte Carlo` method, the interval forecasts are derived from the forecast density.

```
predict(object, ..., n.ahead = 1, conf.lev = 0.95, st.new = NULL,
st.num = NULL, newdata = NULL,
method = c('naive', 'Monte Carlo', 'bootstrap'))
```

The `predict` method returns a list with a class attribute `"vlstarpred"` and two elements: a list denoted with the name `forecasts` containing the predicted values and the interval forecasts for each of the steps ahead, and the matrix with the values of y. The `print` method is applicable to objects of this class and returns the forecasts with upper and lower interval forecasts. The `plot` method draws the time series plots with the interval forecasts in the out-of-sample period.

3 VLSTAR model compared to other linear and nonlinear models

The here applied VLSTAR model is one of the possible ways of modelling nonlinear relationships. Alternatively, nonlinearity in a multivariate framework can be modelled through a Threshold Vector Autoregression (TVAR) or Markov-switching Vector Autoregressive (MSVAR) model. The VLSTAR and the TVAR models are both based on the assumption that the variable that defines the regime-switching is observable, while the MSVAR is mainly based on the assumption that regime-switches are defined by a latent Markov process. When the practitioner has enough information on the factors that drive the dynamics of the dependent variables, using VLSTAR or TVAR models may reduce the uncertainty related to the regimes and may produce more accurate predictions than an MSVAR model (see Hubrich and Teräsvirta, 2013). In other words, the VLSTAR is a model with a continuum of states where the change between a number of regimes is smooth, the TVAR is mostly conceived to analyse the dynamics of variables that switch abruptly between the regimes. The VLSTAR model can be seen as a general version of the TVAR that allows also for the regimes to change smoothly. Indeed, when $\gamma \to \infty$ for each regime, the VLSTAR model becomes a TVAR model with well-defined changes of regimes. Conversely, when $\gamma \to 0$, the model becomes a simple VAR model.

The **starvars** package further differs from the **tsDyn** and the **MSBVAR** by Brandt (2016) packages, which permit the estimation of the TVAR and MSVAR models, since it allows the use of exogenous variables in the estimation set. This is a useful tool since practitioners may control for potential explanatory variables different from lags of the dependent variable to obtain parameter estimates and dependent variables predictions.

4 Example

To illustrate how the R package **starvars** works in practical situations, we present an empirical application with multivariate time series of stock prices. Starting from the prices of $n = 3$ stocks of the tech companies, Amazon, Microsoft and Google, available in the dataset `techprices`, we model the monthly realized covariances assuming that their dynamics can be captured by a flexible specification like the VLSTAR model which nests the linear VAR. First, we construct the $n(n+1)/2$ monthly series of realized covariances and their Cholesky factors which are modelled through VLSTAR. This solves the problem of obtaining positive semidefinite covariance matrices that can be used in finding optimal portfolios. Second, from the estimated VLSTAR, we can compute the forecasts of the monthly realized covariances, see Halbleib-Chiriac and Voev (2011); Bucci et al. (2019); Bucci (2020). In particular, asset returns co-volatilities tend to be higher when bad news is available. From Figure 1, it is clearly observable that co-volatilities explode during periods of market turmoil, like the subprime mortgage crisis in 2007 or the spread of the COronaVIrus Disease 19 (COVID-19) at the beginning of 2020. This explains why co-volatilities exhibit nonlinear behaviour.

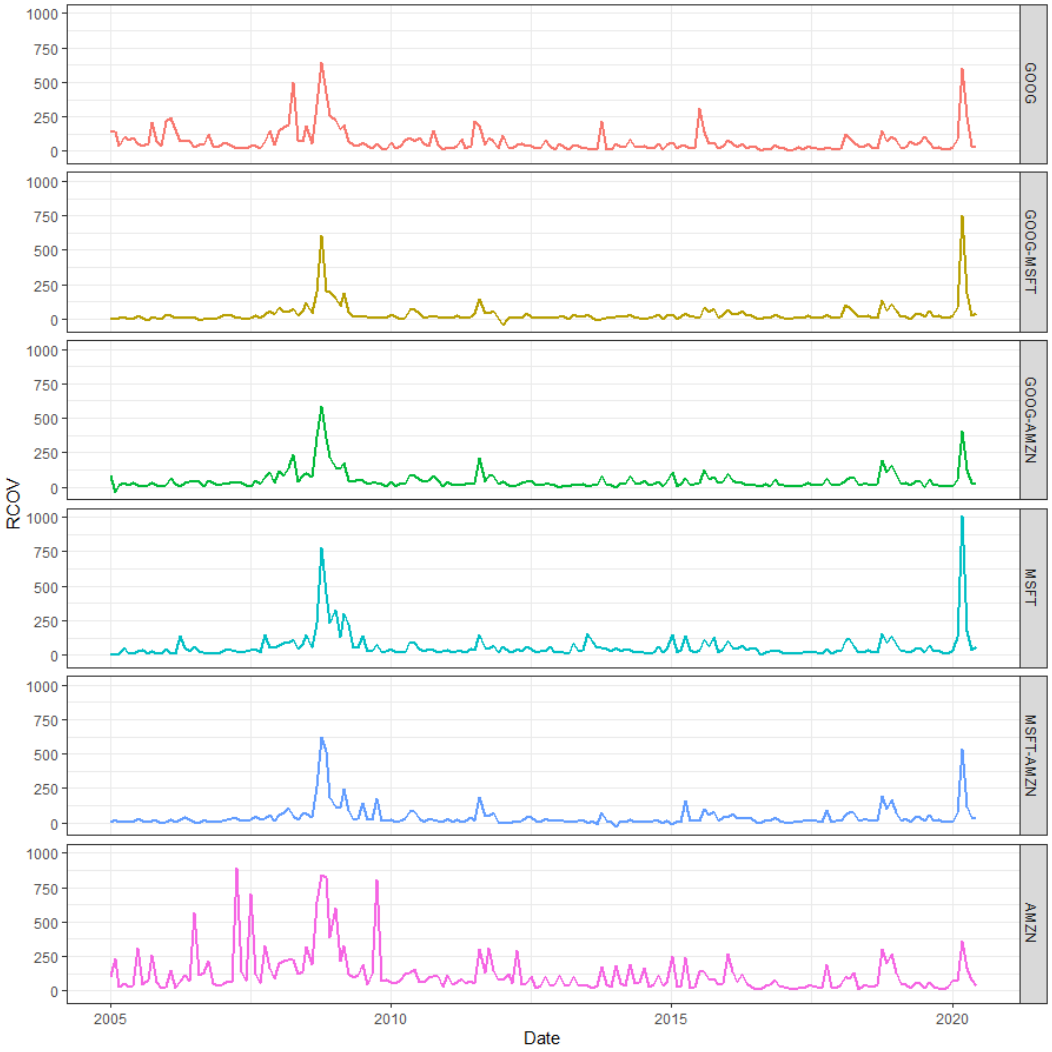


Figure 1: Plots of realized covariances of the stock returns. The panels report the realized variances (one stock symbol, *i.e.* first, third and last panel) and covariances (two symbols, *i.e.* second, third and fifth panel) between the considered stocks. 'GOOG' is the stock symbol of Google, 'MSFT' is the stock symbol of Microsoft, and 'AMZN' is the stock symbol of Amazon. The time series show several peaks during periods of financial market stress such as the sub-prime mortgage crisis and the COVID-19 pandemic in 2021, which may underline a nonlinear behaviour of co-volatilities.

The `techprices` dataset used in this example includes the closing prices from January 1st 2005 to June 16th 2020, for a total of 3,890 observations per series. The dataset can be loaded in the workspace using

```
> data("techprices", package = "starvars")
```

where `techprices` is a $3,890 \times 3$ `xts` object containing the daily prices. As a first step, we calculate the realized covariances of stock returns and their Cholesky factors. Since we have already daily prices, we can only build monthly, quarterly, or yearly realized covariances. To keep the sample of realized covariances quite large, we calculate monthly realized covariances and their Cholesky factors through the code (further discussed in Appendix B):

```
> RCOV <- rcov(techprices, freq = "monthly", make.ret = TRUE, cholesky = TRUE)
```

from which we obtain a list of two elements in the object `RCOV`. We are just interested in the Cholesky factors of the stock returns, thus we save the desired `data.frame` in the object `techchol` with a class `"xts"`.

```
> techchol <- RCOV$'Cholesky Factors'
```

which has dimension $T \times n(n+1)/2$, where $T = 186$ and $n(n+1)/2 = 6$. Therefore, in our example there are $n(n+1)/2 = 6$ dependent variables.

The modelling strategy of a VLSTAR model starts with a test for the time series nonlinearities. As largely explained above, this can be done via the `VLSTARjoint` function. Since no information about which variable should be used as a transition variable is available, we let the linearity test choose among a set of potential variables which are equal to the first lag of the dependent variables. The LM statistics and the related p-value for a given value of `alpha` (set equal to 0.05 by default) and for the chosen transition variable can be obtained simply by running

```
> st <- lag(techchol,1)[-1]
> VLSTARjoint(techchol[-1,], st = st, st.choice = TRUE)

Joint linearity test (Third-order Taylor expansion)
Transition variable chosen: y5
LM = 158.7 ; p-value = 2.0595e-21
Critical value for alpha = 40.646
```

The linearity test indicates the presence of nonlinearity in the data, and that the rejection of the null hypothesis is stronger when the lag of the fifth Cholesky factor, `y5`, is chosen as the transition variable. At this point, the practitioner should assess the presence of common breaks among the time series through the test presented in Appendix A. The test, for a maximum number of breaks equal to 3, is computed as follows.

```
> multiCUMSUM(techchol[-1], max.breaks = 3)
============================================================
Break detection in the covariance structure:
Lambda Omega Break Date 1 Break Date 2 Break Date 3
Break 1  11.10  3.93   2009-04-03
Break 2  21.53  9.64   2009-04-03   2007-12-03
Break 3  12.09  6.03   2009-04-03   2007-12-03   2015-07-03
============================================================
Critical values are 2.69 for Lambda and 1.74 for Omega.
2 Break(s) identified with Lambda
2 Break(s) identified with Omega
```

This function returns significant test statistics for all the breaks for Λ_T and Ω_T, which both identify a number of breaks equal to 2. To keep the model parsimonious, we decide to include a single break and $m = 2$ regimes in our example.

Given that a nonlinear model would be necessary and that at least a single break is present in the multivariate time series, a VLSTAR model can be estimated. Before estimating the parameters, we implement the searching grid algorithm to find starting values of γ and c with 20 potential values each (400 combinations). Specifying `singlecgamma = FALSE` we are supposing that each equation has its own parameters. Once executed the code, a progress bar is shown to inform the user about the completion of the searching grid algorithm.

```
>  starting <- startingVLSTAR(techchol[-1,], p = 1, m = 2, st = st[,5],
+                n.combi = 20, singlecgamma = FALSE, ncores = 4)
```

We employ an NLS estimation, with the lag of the fifth Cholesky factor as s_t, a single lag $p = 1$, two regimes $m = 2$, a number of maximum iterations equal to 30 and a number of cores for parallel computation equal to 4, and we use the starting values found in the previous step of the procedure saved in the `starting` object. Therefore, we show the code used to specify the VLSTAR model as well as the summary output, and the graphic for the equation of the first Cholesky factor, `y1`.

```
>  fit.VLSTAR <- VLSTAR(techchol[-1,], p = 1, m = 2, st = st[,5],
+        method = 'NLS', starting = starting, n.iter = 30, ncores = 4)
> summary(fit.VLSTAR)
> plot(fit.VLSTAR, names = "y1")
Model VLSTAR with 2 regimes
Full sample size: 184
Number of estimated parameters: 108         Multivariate log-likelihood: 2272.663
================================================

Equation y1

Coefficients regime 1
```

```
const        y1        y2        y3        y4        y5        y6
8.108***    0.038     0.135     0.123     0.142  -1.379***    0.330

Coefficients regime 2
const        y1        y2        y3        y4        y5        y6
10.613***  0.411***   -0.067   0.593*** -1.884***   0.669**  1.762***

Gamma: 3.0809         c: 3.1603
AIC: 769.78         BIC: 814.79         LL: -370.89

Equation y2

Coefficients regime 1
const      y1      y2      y3      y4      y5      y6
0.511  -0.019   0.106  0.250**  0.126  -0.005   0.261

Coefficients regime 2
const        y1        y2        y3        y4        y5        y6
6.919***  0.760***   -0.136     0.177* -0.644*** -1.688***  0.613***

Gamma: 866.3921         c: 3.5162
AIC: 545.65         BIC: 590.66         LL: -258.83

Equation y3

Coefficients regime 1
const       y1       y2       y3       y4       y5       y6
1.015*  -0.033    0.053  0.389***   0.003    0.022    0.295

Coefficients regime 2
const        y1        y2        y3        y4        y5        y6
-3.503***  1.419***   -0.123     0.218* -0.580*** -0.895***   -0.425*

Gamma: 110.8034         c: 3.595
AIC: 571.67         BIC: 616.67         LL: -271.83

Equation y4

Coefficients regime 1
const        y1        y2        y3        y4        y5        y6
4.270***   -0.034    -0.046     0.058    0.340** -1.114***    0.096

Coefficients regime 2
const        y1        y2        y3        y4        y5        y6
11.561***   0.127**    0.166.  0.287*** -0.939*** -0.497***  1.117***

Gamma: 1.1841         c: 3.4705
AIC: 496.2         BIC: 541.21         LL: -234.1

Equation y5

Coefficients regime 1
const      y1      y2      y3      y4      y5      y6
0.367  -0.009   0.061   0.096. -0.012  0.200**  0.158

Coefficients regime 2
const        y1        y2        y3        y4        y5        y6
7.756*** -0.695*** -0.337***  0.290*** -0.418***  0.639***  1.269***

Gamma: 100         c: 4.1137
AIC: 351.31         BIC: 396.32         LL: -161.66

Equation y6
```

```
Coefficients regime 1
const        y1        y2        y3        y4        y5        y6
2.693***   -0.005     0.005     0.048     0.120. -0.234***    0.171.

Coefficients regime 2
const        y1        y2        y3        y4        y5        y6
3.648***  0.383***   -0.138*  0.199*** -0.992***   0.178**  0.909***

Gamma: 69.405         c: 3.5824
AIC: 324.3         BIC: 369.31         LL: -148.15
================================================

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

After the execution of the code, a counter with the number of the iteration in the estimation algorithm is shown until convergence or the maximum number of iterations is reached. Using a laptop with an Intel® Core™i5-7200U 2.5GHz processor with 16 GB RAM, the searching grid algorithm takes around 40 seconds to find optimal values of γ and c, while convergence is achieved after 7 iterations taking around 500 seconds (with the package version 1.1.10). The estimation process could take from a few minutes to several hours depending on the complexity of the model. The number of parameters increases with the number of dependent variables, the number of exogenous variables, and the number of regimes, therefore affecting the optimization problem and the convergence time. For example, the estimation of the former example with $m = 3$ regimes takes about an hour and 30 minutes.

The results of the `plot` function on the Equation of `y1` in the `VLSTAR` object are shown in Figure 2. It may be noticed from the last panel of the Figure reporting the logistic function that the assumption of a smoothing regime-switching is realistic.

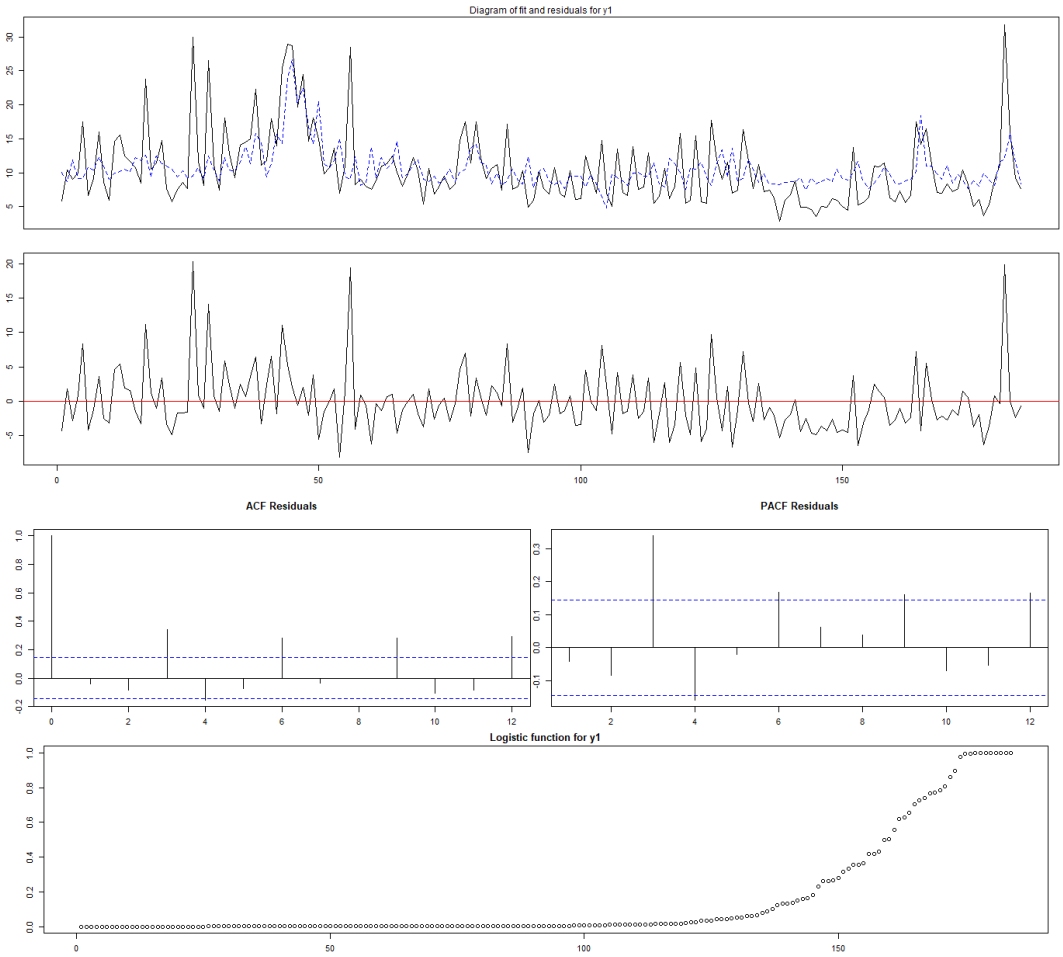


Figure 2: Plots of results from VLSTAR estimation for the Equation of y1. The first panel shows the observed time series (in black) versus the fitted time series (in dashed blue). The second panel shows the residuals and highlights the zero with a red horizontal line. The left side of the third panel reports the autocorrelation function of the residuals, while the right side reports the partial autocorrelation function of the residuals. The fourth panel is about the logistic function that regulates the regime switches. The residual time series of $y1$ seems to show poor autocorrelation, while the regime switches appear to be quite smooth.

Time series models are usually implemented to make out-of-sample predictions. In our package, this is possible through the `predict` method that, applied to objects of class `"VLSTAR"`, returns an object with a class `"vlstarpred"`. When using the `predict` function, the argument `method = 'bootstrap'` specifies that the aforementioned "bootstrap" method has been used to make predictions, while the argument `n.ahead = 2` denotes that two-step-ahead predictions are obtained. The outcome of the `plot` method of the out-of-sample forecasts for the first Cholesky factor is exhibited in Figure 3. The predictions of the Cholesky factors could be used to obtain a semidefinite positive predicted covariance matrix by simply inverting the Cholesky decomposition.

```
> pred.bootstrap <- predict(fit.VLSTAR, n.ahead = 2, st.num = 5, method = 'bootstrap')
> pred.bootstrap
$y1
                   fcst lower 95% upper 95%
Step 1  8.370493  7.283483  9.457503
Step 2 20.916559 12.878648 28.649321

$y2
                   fcst lower 95% upper 95%
Step 1 3.131276  2.540087  3.722465
Step 2 6.188201  4.761677  7.948755

$y3
                   fcst lower 95% upper 95%
Step 1 3.508982  2.874487  4.143478
Step 2 6.631187  4.822495  9.018994

$y4
                   fcst lower 95% upper 95%
Step 1  5.188099  4.671238   5.70496
Step 2 12.483377  8.961486  15.73787

$y5
                   fcst lower 95% upper 95%
Step 1 1.794161  1.445520  2.142802
Step 2 3.293723  2.469695  4.301613

$y6
                   fcst lower 95% upper 95%
Step 1 3.381696  3.057729  3.705664
Step 2 7.258409  6.307594  8.322091


> plot(pred.bootstrap, type = 'single', names = 'y1')
```

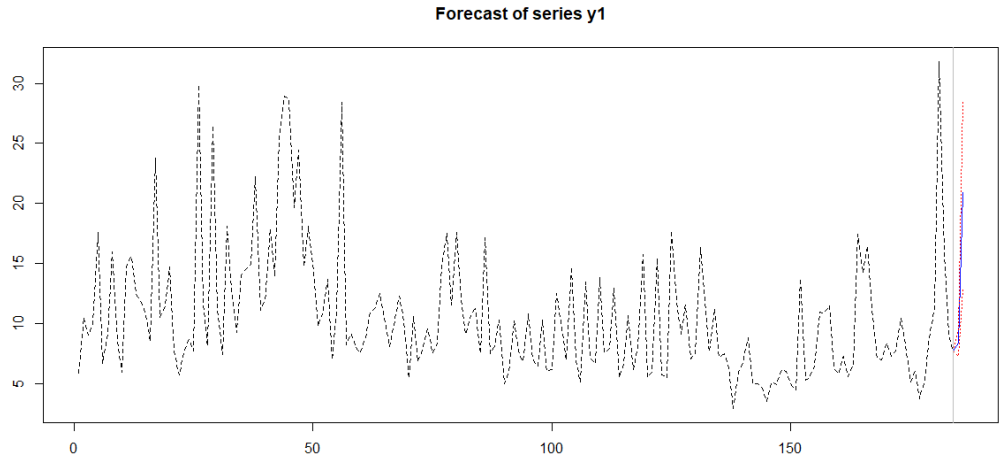


Figure 3: Out-of-sample predictions of time series y1. The plot shows the observed time series in-sample (in dashed black), the two-step ahead out-of-sample predictions (in dashed blue), and their 95% prediction interval (in dashed red). A vertical grey line denotes the end of the in-sample observations. The predictions of time series $y1$ highlight that the prediction interval is extremely tight and that predictions can be nonlinear.

5 Conclusion

This article introduces the R package **starvars** for modelling, estimating, and forecasting a Vector Logistic Smooth Transition Autoregressive (VLSTAR) model. We present the model specification in a general way and illustrate the package usage. In particular, we perform an empirical application using financial data.

The package allows practitioners in many scientific areas to perform their applied research using VLSTAR models in a user-friendly environment. The build-in framework permits to analyse non-linearity of time series and make multi-step-ahead predictions via different methods. Further, the practitioner may use the **starvars** package to obtain realized covariances at several frequencies and the Cholesky decomposition of the related realized covariance matrices.

It should be reminded that the estimation of the parameters in a VLSTAR model strongly depends on the initial values of the parameter of the logistic. We have observed that sometimes the algorithm underlying the automatic grid search may lead to unrealistic estimates of the logistic parameters and, consequently, to not consistent estimates of coefficients. Moreover, the computational time, when using more than two regimes, might be compromised by a large number of coefficients and a possible local minimum may be found by the maximization of the log-likelihood. Thus, the suggestion is to use a limited number of regimes to keep the model as parsimonious as possible.

The code of the package **starvars** may be improved by using a different transition variable for each equation or by allowing the estimates of a univariate model. However, in both cases, the estimation would be reduced to a univariate model for each equation and there are already packages able to do this.

6 Availability

The here presented package is written using S4 classes and provides methodology such as `coef`, `plot`, `AIC`, `BIC`, `logLik`, `summary` and `print` to analyze the results. The R package **starvars** is available from the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/package=starvars and on GitHub at https://github.com/andbucci/starvars.

Bibliography

T. G. Andersen, T. Bollerslev, F. X. Diebold, and H. Ebens. The distribution of realized stock return volatility. Journal of Financial Economics, 61:43–76, 2001. doi: 10.1016/S0304-405X(01)00055-1. [p224]

T. G. Andersen, T. Bollerslev, F. X. Diebold, and P. Labys. Modeling and Forecasting Realized Volatility. Econometrica, 71:579–625, 2003. doi: 10.1111/1468-0262.00418. [p224]

A. Aue, S. Hörmann, L. Horváth, and M. Reimherr. Break detection in the covariance structure of multivariate time series models. Annals of Statistics, 37(6B):4046–4087, 12 2009. doi: 10.1214/09-AOS707. [p224]

J. Bai and P. Perron. Estimating and Testing Linear Models With Multiple Structural Changes. Econometrica, 66:47–78, 1998. doi: 10.2307/2998540. [p224]

J. Bai and P. Perron. Computation and Analysis of Multiple Structural Change Models. Journal of Applied Econometrics, 18:1–22, 2003. doi: 10.1002/jae.659. [p224]

J. Bai, R. Lumsdaine, and J. Stock. Testing for and Dating Common Breaks in Multivariate Time Series. Review of Economic Studies, 65(3):395–432, 1998. doi: 10.1111/1467-937X.00051. [p224]

M. Balcilar. RSTAR: A Package for Smooth Transition Autoregressive (STAR) Modeling Using R, 2016. doi: 10.13140/RG.2.1.3031.7841. [p208]

M. Barassi, L. Horváth, and Y. Zhao. Change-Point Detection in the Conditional Correlation Structure of Multivariate Volatility Models. Journal of Business & Economic Statistics, 38(2):340–349, 2020. doi: 10.1080/07350015.2018.1505630. [p224]

O. E. Barndorff-Nielsen and N. Shephard. Econometric analysis of realised volatility and its use in estimating stochastic volatility models. Journal of the Royal Statistical Society, 64(2):253–280, 2002. URL http://www.jstor.org/stable/3088799. [p224]

R. Becker, A. Clements, and R. O'Neill. A Cholesky-MIDAS model for predicting stock portfolio volatility. Technical report, 2010. URL http://www.ncer.edu.au/papers/documents/WPNo60.pdf. Working paper, Centre for Growth and Business Cycle Research Discussion Paper Series. [p225]

P. Brandt. MSBVAR: Markov-Switching, Bayesian, Vector Autoregression Models, 2016. URL https://CRAN.R-project.org/package=MSBVAR. R package version 0.9-3. [p208, 214]

A. Bucci. Cholesky-ANN models for predicting multivariate realized volatility. Journal of Forecasting, 39(6):865–876, 2020. doi: 10.1002/for.2664. [p214, 225]

A. Bucci, G. Palomba, and E. Rossi. Does macroeconomics help in predicting stock markets volatility comovements? a nonlinear approach, 2019. URL http://docs.dises.univpm.it/web/quaderni/pdf/440.pdf. Working Paper 440, Università Politecnica delle Marche, Dipartimento di Scienze Economiche e Sociali. [p214, 225]

A. Bucci, G. Palomba, and E. Rossi. starvars: Vector Logistic Smooth Transition Models Estimation and Prediction, 2022. URL https://CRAN.R-project.org/package=starvars. R package version 1.1.10. [p208]

G. Caggiano, E. Castelnuovo, V. Colombo, and G. Nodari. Estimating fiscal multipliers: News from a non-linear world. The Economic Journal, 125(584):746–776, 2015. doi: 10.1111/ecoj.12263. [p208]

M. Camacho. Vector smooth transition regression models for US GDP and the composite index of leading indicators. Journal of Forecasting, 23:173–196, 2004. doi: 10.1002/for.912. [p208]

A. F. Di Narzo, J. L. Aznarte, M. Stigler, and H. Tsung-wu. tsDyn: Nonlinear Time Series Models with Regime Switching, 2020. URL https://CRAN.R-project.org/package=tsDyn. R package version 10-1.2. [p208]

F. Diebold and J. Nason. Nonparametric exchange rate prediction? Journal of International Economics, 28(3):315–332, 1990. doi: 10.1016/0022-1996(90)90006-8. [p212]

R. Halbleib-Chiriac and V. Voev. Modelling and Forecasting Multivariate Realized Volatility. Journal of Applied Econometrics, 26:922–947, 2011. doi: 10.1002/jae.1152. [p214, 225]

K. Hubrich and T. Teräsvirta. Thresholds and Smooth Transitions in Vector Autoregressive Models, 2013. CREATES Research Paper, Aarhus University. doi: 10.1108/S0731-9053(2013)0000031008. [p208, 210, 213, 214]

A. B. Kock and T. Teräsvirta. Forecasting with nonlinear time series models, 2011. CREATES Research Paper, Aarhus University. doi: 10.2139/ssrn.1531092. [p212, 213]

S. Lundbergh and T. Teräsvirta. Forecasting with Smooth Transition Autoregressive Models, chapter 21, pages 485–509. John Wiley & Sons, Ltd, 2007. ISBN 9780470996430. doi: 10.1002/9780470996430.ch21. [p213]

R. Luukkonen, P. Saikkonen, and T. Teräsvirta. Testing Linearity Against Smooth Transition Autoregressive Models. Biometrika, 75:491–499, 1988. doi: 10.2307/2336599. [p210]

R. E. Quandt. The estimation of the parameters of a linear regression system obeying two separate regimes. Journal of American Statistical Association, 53:873–880, 1958. doi: 10.2307/2281957. [p208]

R. E. Quandt. Tests of the hypothesis that a linear regressions system obeys two different regimes. Journal of American Statistical Association, 55:324–330, 1960. doi: 10.2307/2281745. [p208]

P. Rothman, D. van Dijk, and P. H. Franses. A multivariate star analysis of the relationship between money and output. Macroeconomic Dynamics, 5:506–532, 2001. doi: 10.1017/S1365100500000444. [p208]

T. Teräsvirta. Specification, Estimation and Evaluation of Smooth Transition Autoregressive Models. Journal of American Statistical Association, 89:208–218, 1994. doi: 10.2307/2291217. [p208, 210]

T. Teräsvirta and K. S. Lim. Threshold autoregression, limit cycles and cyclical data. Journal of the Royal Statistical Society Series B, 42:245–292, 1980. doi: 10.1111/j.2517-6161.1980.tb01126.x. [p208]

T. Teräsvirta and Y. Yang. Specification, estimation and evaluation of vector smooth transition autoregressive models with applications. CREATES Research Paper 2014-8, 2014a. doi: 10.1.1/698.2255. [p208, 209, 211]

T. Teräsvirta and Y. Yang. Linearity and misspecification tests for vector smooth transition regression models. CREATES Research Paper 2014-4, 2014b. URL https://pure.au.dk/ws/files/90929819/rp14_04.pdf. [p210]

T. Teräsvirta, D. Tjøstheim, and C. W. Granger. Modelling Nonlinear Economic Time Series. Oxford University Press, 2010. doi: 10.1093/acprof:oso/9780199587148.001.0001. [p210, 213]

H. Tong. On a Threshold Model, Pattern Recognition and Signal Processing, pages 575–586. Sijthoff & Nordhoff, Netherlands, 1978. URL http://eprints.lse.ac.uk/id/eprint/19500. [p208]

R. S. Tsay. Testing and modeling multivariate threshold models. Journal of American Statistical Association, 93:1188–1202, 1998. doi: 10.2307/2669861. [p208]

7 Appendix A: Testing for common breaks

If the linearity hypothesis is rejected, the researcher should determine the number of regimes of the dependent variable. To this end, the procedure introduced by Bai and Perron (1998, 2003) may be implemented. In presence of multivariate time series, it may happen that sudden shocks, such as market crashes, financial crises, or interventions of policymakers, result in a structural break in the mean of the observed time series (see Bai et al., 1998). At the same time, the interest of the researcher may be directed to changes in the structure of the conditional correlations (see Barassi et al., 2020; Aue et al., 2009). To detect the presence of structural breaks in the co-movements of the n time series, Aue, Hörmann, Horváth, and Reimherr (2009) introduced a test on the structure of the covariances. Here, we attempt to summarize the procedure[1].

Let $(y_t : t \in \mathbb{Z})$ be a sequence of n time series, with $E[y_t] = \mu$ and $E[|y_t|^2] < \infty$, where $|\cdot|$ denotes the Euclidean norm in $\Re^n$, then the null hypothesis in a test for structural breaks in the co-volatilities process is given by

$$H_0 \colon Cov(y_1) = \ldots = Cov(y_T)$$

where T is the number of observations. This means that the covariances are constant over the observed period. A common alternative hypothesis would be that there is at least one change in the covariance structure which corresponds to the presence of at least one common break.

Provided that $E[y_t] = 0$, the test statistic is based on the constancy of the expected values $E[vech(y_t y_t')]$ for $t = 1, \ldots, T$ under H_0. As a consequence, from the estimates of $E[vech(y_t y_t')]$ on j observations (with $j < T$), a traditional cumulative sum (CUSUM) statistic can be constructed as

$$S_j = \frac{1}{\sqrt{T}} \left(\sum_{t=1}^{j} vech[y_t y_t'] - \frac{j}{T} \sum_{t=1}^{T} vech[y_t y_t'] \right), \text{with } j = 1, \ldots, T. \tag{16}$$

Let $\tilde{y}_t = y_t - \overline{y}_T$, where $\overline{y}_T = \frac{1}{T} \sum_{t=1}^{T} y_t$, if the zero mean assumption does not hold, *i.e.* $E[y_t] \neq 0$, then S_j can be replaced by

$$\tilde{S}_j = \frac{1}{\sqrt{T}} \left(\sum_{t=1}^{j} vech[\tilde{y}_t \tilde{y}_t'] - \frac{j}{T} \sum_{t=1}^{T} vech[\tilde{y}_t \tilde{y}_t'] \right), \text{with } j = 1, \ldots, T. \tag{17}$$

Given the long-run covariance estimator $\hat{\Sigma}_T$, the test statistics are

$$\Lambda_T = \max_{1 \leq j \leq T} S_j' \hat{\Sigma}_T^{-1} S_j \text{ and } \Omega_T = \frac{1}{T} \sum_{j=1}^{T} S_j' \hat{\Sigma}_T^{-1} S_j \tag{18}$$

as well as

$$\tilde{\Lambda}_T = \max_{1 \leq j \leq T} \tilde{S}_j' \hat{\Sigma}_T^{-1} \tilde{S}_j \text{ and } \tilde{\Omega}_T = \frac{1}{T} \sum_{j=1}^{T} \tilde{S}_j' \hat{\Sigma}_T^{-1} \tilde{S}_j.$$

For the critical values of these statistics, it should be referred to Aue, Hörmann, Horváth, and Reimherr (2009).

Once the null hypothesis can be rejected, the researcher should find the location of both the breakpoint and the breakpoint fraction θ whose estimation is given by

$$\hat{\theta} = \frac{1}{T} \arg\max_{1 \leq j \leq T} S_j' \hat{\Sigma}_T^{-1} S_j. \tag{19}$$

This can be repeated for each partition of the entire sample to obtain the optimal number and location of common breaks. On the basis of what is found with the test on common breaks, the number of regimes of the VLSTAR model can be assessed and parameters estimation can be performed.

8 Appendix B: Realized covariances construction

Along with the specification of a VLSTAR model, the R package **starvars** allows the user to calculate a non-parametric measure of volatility in the multivariate framework, such as the realized volatility (see Andersen et al., 2001, 2003; Barndorff-Nielsen and Shephard, 2002, for the theoretical fundamentals). Given a vector of stock returns, r_τ sampled at a given frequency, τ, the realized covariance matrix,

[1] See the original paper by Aue, Hörmann, Horváth, and Reimherr (2009) for the technical details.

RC_t observed at a lower frequency t is simply given by

$$RC_t = \sum_{\tau=1}^{N_t} r_\tau r'_\tau \tag{20}$$

where N_t is the number of observations in the t-th period and $t = 1, \ldots, T$.

The function `rcov` in the package **starvars** returns the lower triangular of RC_t starting both from stock prices or returns, and to calculate it for different frequencies.

```
rcov(data, freq = c('daily', 'monthly', 'quarterly', 'yearly'),
make.ret = TRUE, cholesky = FALSE)
```

The function consists of several arguments. An object of class `"xts"` with the values of stock prices or returns on which the realized covariances should be calculated. The frequency of t, which could be `daily`, `monthly`, `quarterly` or `yearly`. The boolean argument `make.ret` denotes whether the data passed as input in the argument `data` should be converted to returns, if `TRUE` the returns are calculated. Finally, since a wide strand of the literature relies on the Cholesky factors of RC_t to make inference or predictions (see Becker, Clements, and O'Neill, 2010; Halbleib-Chiriac and Voev, 2011; Bucci, Palomba, and Rossi, 2019; Bucci, 2020, for example), the function also allows the user to calculate the Cholesky factors, L_t, such that

$$RC_t = L_t L'_t.$$

This can be done by setting the argument `cholesky` equal to `TRUE`. If `make.ret` is set equal to `TRUE`, the output of the function `rcov` contains an element of class `"xts"` with the returns.

When `cholesky = TRUE`, the output of the `rcov` function is a list containing the $T \times n(n+1)/2$ `xts` object from the vectorization of the realized covariance matrices, given by $vech(RC_t)$, and the $T \times n(n+1)/2$ of the vectorization of L_t, given by $vech(L_t)$, otherwise it includes only the series of realized covariances.

Andrea Bucci
Department of Economics, Università degli Studi G. d'Annunzio Chieti-Pescara
Viale Pindaro 42, Pescara
Italy
https://orcid.org/0000-0001-9872-9761
andrea.bucci@unich.it

Giulio Palomba
Department of Economics and Social Sciences, Università Politecnica delle Marche
Piazzale Martelli 8, Ancona
Italy
g.palomba@staff.univpm.it

Eduardo Rossi
Department of Economics and Management, University of Pavia
Via S. Felice Al Monastero 7, Pavia
Italy
https://orcid.org/0000-0003-3597-8060
eduros04@unipv.it

fairmodels: a Flexible Tool for Bias Detection, Visualization, and Mitigation in Binary Classification Models

by Jakub Wiśniewski and Przemysław Biecek

Abstract Machine learning decision systems are becoming omnipresent in our lives. From dating apps to rating loan seekers, algorithms affect both our well-being and future. Typically, however, these systems are not infallible. Moreover, complex predictive models are eager to learn social biases present in historical data that may increase discrimination. If we want to create models responsibly, we need tools for in-depth validation of models also from potential discrimination. This article introduces an R package **fairmodels** that helps to validate fairness and eliminate bias in binary classification models quickly and flexibly. The **fairmodels** package offers a model-agnostic approach to bias detection, visualization, and mitigation. The implemented functions and fairness metrics enable model fairness validation from different perspectives. In addition, the package includes a series of methods for bias mitigation that aim to diminish the discrimination in the model. The package is designed to examine a single model and facilitate comparisons between multiple models.

1 Introduction

Responsible machine learning and, in particular, fairness is gaining attention within the machine learning community. This is because predictive algorithms are becoming more and more decisive and influential in our lives. This impact could be less or more significant in areas ranging from user feeds on social platforms, displayed ads, and recommendations at an online store to loan decisions, social scoring, and facial recognition systems used by police and authorities. Sometimes it leads to automated systems that learn some undesired bias preserved in data for some historical reason. Whether seeking a job (Lahoti et al., 2019) or having one's data processed by court systems (Angwin et al., 2016), sensitive attributes such as sex, race, religion, ethnicity, etc., might play a significant role in the decision. Even if such variables are not directly included in the model, they are often captured by proxy variables such as zip code (a proxy for the race and wealth), purchased products (a proxy for gender and age), eye colour (a proxy for ethnicity). As one would expect, they can give an unfair advantage to a privileged group. Discrimination takes the form of more favorable predictions or higher accuracy for a privileged group. For example, some popular commercial gender classifiers were found to perform the worst on darker females (Buolamwini and Gebru, 2018). From now on, such unfair and harmful decisions towards people with specific sensitive attributes will be called biased.

The list of protected attributes may depend on the region and domain for which the model is built. For example, the European Union law is summarized in the Handbook on European non-discrimination law European Union Agency for Fundamental Rights and Council of Europe (2018), which lists the following protected attributes that cannot be the basis for inferior treatment: sex, gender identity, sexual orientation, disability, age, race, ethnicity, nationality or national origin, religion or belief, social origin, birth, and property, language, political or other opinions. This list, though long, does not include all potentially relevant items, e.g. in the USA, a protected attribute is also pregnancy, the status of a war veteran, or genetic information.

While there are historical and economic reasons for this to happen, such decisions are unacceptable in society, where nobody should have an unfair advantage. The problem is not simple, especially when the only criterion set for the system is performance. We observe a trade-off between accuracy and fairness in some cases where lower discrimination leads to lower performance (Kamiran and Calders, 2011). Sometimes labels, which are considered ground truth, might also be biased (Wick et al., 2019), and when controlling for that bias, the performance and fairness might improve simultaneously. However fairness is not a concept that a single number can summarize, so most of the time, when we want to improve fairness from one perspective, it becomes worse in another (Barocas et al., 2019).

The bias in machine learning systems has potentially many different sources. Mehrabi et al. (2019) categorized bias into its types like historical bias, where unfairness is already embedded into the data reflecting the world, observer bias, sampling bias, ranking and social biases, and many more. That shows how many dangers are potentially hidden in the data itself. Whether one would like to act on it or not, it is essential to detect bias and make well-informed decisions whose consequences could potentially harm many groups of people. Repercussions of such systems can be unpredictable. As argued by Barocas et al. (2019), machine learning systems can even aggravate the disparities between groups, which is called by the authors' feedback loops. Sometimes the risk of potential harm resulting

from the usage of such systems is high. This was noticed, for example, by the Council of Europe that wrote the set of guidelines where it states that the usage of facial recognition for the sake of determining a person's sex, age, origin, or even emotions should be mostly prohibited (Council of Europe, 2021).

Not every difference in treatment is discrimination. Cirillo et al. (2020) presents examples of desirable and undesirable biases based on the medical domain. For example, in the case of cardiovascular diseases, documented medical knowledge indicates that different treatments are more effective for different genders. So different treatment regimens according to medical knowledge are examples of desirable bias. Later in this paper, we present tools to identify differences between groups defined by some protected attribute but note that this does not automatically mean that there is discrimination.

We would also like to point out that focusing on the machine learning model may not be enough in some cases, and sometimes the design of the data acquisition and/or annotation cause the model to be biased (Barocas et al., 2019).

Related work

Assembling predictive models is getting easier nowadays. Packages like **h2o** (H2O.ai, 2017) provide AutoML frameworks where non-experts can train quickly accurate models without deep domain knowledge. Model validation should also be that simple. Yet this is not the case. There are still very few tools to support the fairness diagnostics of the model.

Two main kinds of fairness are a concern to multiple stakeholders. These are group and individual fairness. The first one concerns groups of people with the same protected attributes (gender, race, etc.). It focuses on measuring if these groups are treated similarly by the model. The second one is focused on the individual. It is most intuitively defined as treating similar individuals similarly (Dwork et al., 2012). Both concepts are sometimes considered to conflict with each other, but they don't need to be if we factor in certain assumptions, such as whether the disparities are due to personal choices or unjust structures (Binns, 2020).

Several frameworks have emerged for Python to verify various fairness criteria, the most popular are **aif360** (Bellamy et al., 2018), **fairlearn** (Bird et al., 2020), or **aequitas** (Saleiro et al., 2018). They have various features for detecting, visualization, and mitigating bias in machine learning models.

For the R language, until recently, the only available tool was the **fairness** (Kozodoi and V. Varga, 2021) package which compares various fairness metrics for specified subgroups. The **fairness** package is very helpful, but it lacks some features. For example, it does not allow comparing the machine learning models and aggregating fairness metrics to facilitate the visualization. Still, most of all, it does not give a quick verdict on whether a model is fair or not. Package **fairadapt** aims at removing bias from machine learning models by implementing pre-processing procedure described in Plečko and Meinshausen (2019). Our package tries to combine the detection and mitigation processes. It encourages the user to experiment with the bias, try different mitigation methods and compare results. The package **fairmodels** not only allows for that comparison between models and multiple exposed groups of people, but it gives direct feedback if the model is fair or not (more on that in the next section). Our package also equips the user with a so-called `fairness_object`, an object aggregating possibly many models, information about data, and fairness metrics. `fairness_object` can later be transformed into many other objects that can facilitate the visualization of metrics and models from different perspectives. If a model does not meet fairness criteria, various pre-processing and post-processing bias mitigation algorithms are implemented and ready to use. It aims to be a complete tool for dealing with discriminatory models in a group fairness setting.

In particular, in the following sections, we show how to use this package to address four key questions: *How to measure bias? How to detect bias? How to visualize bias?* and *How to mitigate bias?*

It is important to remember that fairness is not a binary concept that can be unambiguously defined, and there is no silver bullet that will make any model fair. The presented tools allow for fairness exploratory analysis, thanks to which we will be able to detect differences in the behavior of the model for different protected groups. But such analysis will not guarantee that all possible fairness problems have been detected. Also, fairness analysis is only one of a wide range of techniques for Explanatory Model Analysis (Biecek and Burzykowski, 2021). Like other explanatory tools, it should be used with caution and awareness.

2 Measuring and detecting bias

In model fairness analysis, a distinction is often made between group fairness and individual fairness analysis. The former is defined by the equality of certain statistics determined on protected subgroups,

$A = a$	$Y = 1$	$Y = 0$	
$\hat{Y} = 1$	$TP_a = P(Y = 1, \hat{Y} = 1 \mid A = a)$	$FP_a = P(Y = 0, \hat{Y} = 1 \mid A = a)$	$P(\hat{Y} = 1 \mid A = a)$
$\hat{Y} = 0$	$FN_a = P(Y = 1, \hat{Y} = 0 \mid A = a)$	$TN_a = P(Y = 0, \hat{Y} = 0 \mid A = a)$	$P(\hat{Y} = 0 \mid A = a)$
	$P(Y = 1 \mid A = a)$	$P(Y = 0 \mid A = a)$	

Figure 1: Summary of possible model outcomes for subpopulation $A = a$. We assume that outcome $Y = 1$ is favourable.

and we focus on this approach in this section. We write more about the latter later in this paper.

Fairness metrics

Machine learning models, just like human-based decisions, can be biased against observations related to people with certain sensitive attributes, which are also called protected groups. This is because they consist of subgroups - people who share the same sensitive attribute, like gender, race, or other features.

To address this problem, we need first to introduce fairness criteria. Following Barocas et al. (2019), we will present these criteria based on the following notation.

- Let $A \in \{a, b, ...\}$ mean protected group and values $A \neq a$ denote membership to unprivileged subgroups while $A = a$ membership to privileged subgroup. To simplify the notation, we will treat this as a binary variable (so $A = b$ will denote membership to unprivileged subgroup), but all results hold if A has a larger number of groups.

- Let $Y \in \{0, 1\}$ be a binary label (binary target = binary classification) where 1 is preferred, favorable outcome.
- Let $R \in [0, 1]$ be a probabilistic response of the model, and $\hat{Y} \in \{0, 1\}$ is the binarised model response, so $\hat{Y} = 1$ when $R \geq 0.5$, otherwise $\hat{Y} = 0$.

Figure 1 summarizes possible situations for the subgroup $A = a$. We can draw up the same table for each of the subgroups.

According to Barocas et al. (2019) most discrimination criteria can be derived as tests that validate the following probabilistic definitions:

- Independence, i.e. $R \perp A$,
- Separation, i.e. $R \perp A \mid Y$,
- Sufficiency, i.e. $Y \perp A \mid R$.

Those criteria and their relaxations might be expressed via different metrics based on a confusion matrix for a certain subgroup. To check if those fairness criteria are addressed, we propose checking five metrics among privileged group (a) and unprivileged group (b):

- Statistical parity: $P(\hat{Y} = 1 \mid A = a) = P(\hat{Y} = 1 \mid A = b)$. Statistical parity (STP) ensures that fractions of assigned positive labels are the same in subgroups. It is equivalent of Independence (Dwork et al., 2012). In other words, the values in the last column of Figure 1 are the same for each subgroup.
- Equal opportunity: $P(\hat{Y} = 1 \mid A = a, Y = 1) = P(\hat{Y} = 1 \mid A = b, Y = 1)$. Checks if classifier has equal True Positive Rate (TPR) for each subgroup. In other words, the column normalized values in the second column of Figure 1 are the same for each subgroup. It is a relaxation of Separation (Hardt et al., 2016).
- Predictive parity: $P(Y = 1 \mid A = a, \hat{Y} = 1) = P(Y = 1 \mid A = b, \hat{Y} = 1)$. Measures if a model has equal Positive Predictive Value (PPV) for each subgroup. In other words, the row normalized values in the second row of Figure 1 are the same for each subgroup. It is relaxation of Sufficiency (Chouldechova, 2016).
- Predictive equality: $P(\hat{Y} = 1 \mid A = a, Y = 0) = P(\hat{Y} = 1 \mid A = b, Y = 0)$. Warrants that classifiers have equal False Positive Rate (FPR) for each subgroup. In other words, the column normalized values in the third column of Figure 1 are the same for each subgroup. It is relaxation of Separation (Corbett-Davies et al., 2017).
- (Overall) Accuracy equality: $P(\hat{Y} = Y \mid A = a) = P(\hat{Y} = Y \mid A = b)$. Makes sure that models have the same Accuracy (ACC) for each subgroup. (Berk et al., 2017)

The reader should note that if the classifier passes Equal opportunity and Predictive equality, it also passes Equalized Odds (Hardt et al., 2016), which is equivalent to Separation criteria.

Let us illustrate the intuition behind Independence, Separation, and Sufficiency criteria using the well-known example of the COMPAS model for estimating recidivism risk. Fulfilling the Independence criterion means that the rate of sentenced prisoners should be equal in each subpopulation. It can be said that such an approach is fair from society's perspective.

Fulfilling the Separation criterion means that the fraction of innocents/guilty sentenced should be equal in subgroups. Such an approach is fair from the prisoner's perspective. The reasoning is the following: *"If I am innocent, I should have the same chance of acquittal regardless of sub-population"*. This was the expectation presented by the ProPublica Foundation in their study.

Meeting the Sufficiency criterion means that there should be an equal fraction of innocents among the convicted, similarly, for the non-convicted. This approach is fair from the judge's perspective. The reasoning is the following: *"If I convicted someone, he should have the same chance of being innocent regardless of the sub-population"*. This approach is presented by the company developing the COMPAS model, Northpointe. Unfortunately, as we have already written, it is not possible to meet all these criteria at the same time.

While defining the metrics above, we assumed only two subgroups. This was done to facilitate notation, but there might be more unprivileged subgroups. A perfectly fair model would pass all criteria for each subgroup (Barocas et al., 2019).

Not all fairness metrics are equally important in all cases. The metrics above aim to give a more holistic view into the fairness of the machine learning model. Practitioners informed in the domain may consider only those metrics that are relevant and beneficial from their point of view. For example, in Kozodoi et al. (2021) in the fair credit scoring use case, the authors concluded that the separation is the most suitable non-discrimination criteria. More general instructions can also be found in European Union Agency for Fundamental Rights (2018), along with examples of protected attributes. Sometimes, however, non-technical solutions to fairness problems might be beneficial. Note that group fairness metrics will discover not all types of unfairness, and the end-user should decide whether a model is acceptable in terms of bias or not.

However tempting it is to think that all the criteria described above can be met at the same time, unfortunately, this is not possible. Barocas et al. (2019) shows that, apart from a few hypothetical situations, no two of *{Independence, Separation, Sufficiency}* can be fulfilled simultaneously. So we are left balancing between the degree of imbalance of the different criteria or deciding to control only one criterion.

Acceptable amount of bias

It would be hard for any classifier to maintain the same relations between subgroups. That is why some margins around the perfect agreement are needed. To address this issue, we accepted the four-fifths rule (Code of Federal Regulations, 1978) as the benchmark for discrimination rate, which states that *"A selection rate for any race, sex, or ethnic group which is less than four-fifths ($\frac{4}{5}$) (or eighty percent) of the rate for the group with the highest rate will generally be regarded by the Federal enforcement agencies as evidence of adverse impact[...]."* The selection rate is originally represented by statistical parity, but we adopted this rule to define acceptable rates between subgroups for all metrics. There are a few caveats to the preceding citation concerning the size of the sample and the boundary itself. Nevertheless, the four-fifths rule is an excellent guideline to adhere to. In the implementation, this boundary is represented by ε, and it is adjustable by the user, but the default value will be 0.8. This rule is often used, but users should check if the fairness criteria should be set differently in each case.

Let $\varepsilon > 0$ be the acceptable amount of a bias. In this article, we would say that the model is not discriminatory for a particular metric if the ratio between every unprivileged $b, c, ...$ and privileged subgroup a is within $(\varepsilon, \frac{1}{\varepsilon})$. The common choice for the epsilon is 0.8, which corresponds to the four-fifths rule. For example, for the metric Statistical Parity (STP), a model would be ε-non-discriminatory for privileged subgroup a if it satisfies.

$$\forall_{b \in A \setminus \{a\}} \quad \varepsilon < STP_{ratio} = \frac{STP_b}{STP_a} < \frac{1}{\varepsilon}. \tag{1}$$

Evaluating fairness

The main function in the **fairmodels** package is `fairness_check`. It returns `fairness_object`, which can be visualized or processed by other functions. This will be further explained in the "Structure" section. When calling `fairness_check` for the first time, the following three arguments are mandatory:

- `explainer` - an object that combines model and data that gives a unified interface for predictions. It is a wrapper over a model created with the **DALEX** (Biecek, 2018) package.
- `protected` - a factor, vector containing sensitive attributes (protected group). It does not need to be binary. Instead, each level denotes a distinct subgroup. The most common examples are gender, race, nationality, etc.
- `privileged` - a character/factor denoting a level in the protected vector which is suspected to be the most privileged one.

Example

In the following example, we are using *German Credit Data* dataset (Dua and Graff, 2017). In the dataset, there is information about people like age, sex, purpose, credit amount, etc. For each person, there is a risk assessed with taking credit, either good or bad. Therefore, it will be a target variable. We will train the model on the whole dataset and then measure fairness metrics to facilitate the notation (as opposed to training and testing on different subsets, which is also possible and advisable).

First, we create a model. Let's start with logistic regression.

```
library("fairmodels")
data("german")

lm_model <- glm(Risk~., data = german, family = binomial(link = "logit"))
```


```
library("fairmodels")
data("german")

lm_model <- glm(Risk~., data = german, family = binomial(link = "logit"))
```


Then, create a wrapper that unifies the model interface.

```
library("DALEX")

y_numeric <- as.numeric(german$Risk) -1
explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)
```

Now we are ready to calculate and plot the fairness checks. Resulting plot is presented in Figure 2.

```
fobject <- fairness_check(explainer_lm,
                protected = german$Sex, privileged = "male",
                verbose = FALSE)

plot(fobject)
```

For a quick assessment, if a model passes fairness criteria, the object created with `fairness_check()` might be summarized with the `print()` function. Total loss is the sum of all fairness metrics. See equation (3) for more details.

```
print(fobject, colorize = FALSE)

#>
#> Fairness check for models: lm
#>
#> lm passes 4/5 metrics
#> Total loss :  0.6153324
```

In this example, fairness criteria are satisfied in all but one metric. The logistic regression model has a lower false-positive rate (FP/(FP+TN))) in the unprivileged group than in the privileged group. It exceeds the acceptable limit set by ε. Thus it does not satisfy the Predictive Equality ratio criteria.

More detailed visualizations are available, like *Metric scores* plot. It might be helpful to understand the intuition behind the *Fairness check* plot presented above. See an example in Figure 3. This plot might be a good first point for understanding the *Fairness check* plot. In fact, checks can be directly derived from the *Metric scores* plot. To do this, we need to divide the score denoted by the dot with the score denoted by the vertical line. This way, we obtain a value indicated by the height of the barplot. The orientation of the barplot depends on whether the value is bigger or lower than 1. Intuitively the longer the horizontal line in the figure below (the one connecting the dot with the vertical line) is, the longer the bar will be in *Fairness check* plot. If the scores of privileged and unprivileged subgroups are the same, then the bar will start from 1 and point to 1, so it will have a height equal to 0.

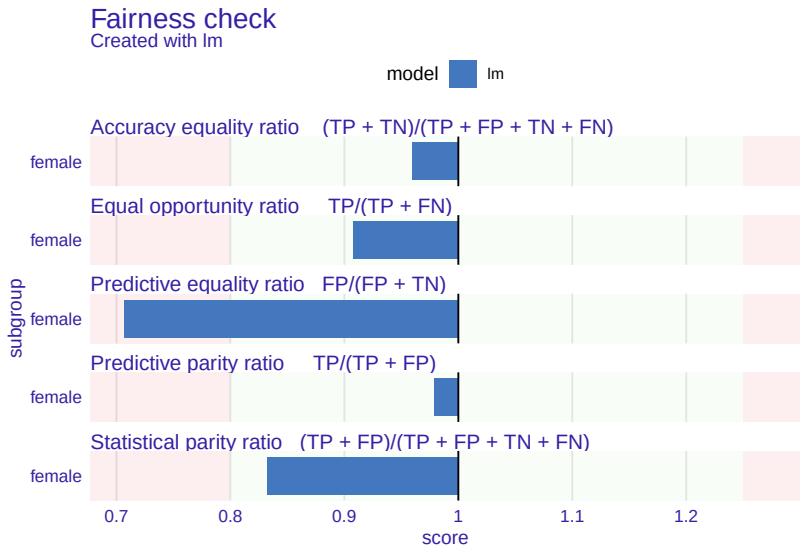


Figure 2: The Fairness Check plot summarises the ratio of fairness measures between unprivileged and privileged subgroups. The light green areas correspond to values within $(\varepsilon, \frac{1}{\varepsilon})$ and signify an acceptable difference in fairness metrics. They are bounded by red rectangles indicating values that do not meet the $4/5$ rule. Fairness metrics names are given along the formulas used to calculate the score in some subgroups to facilitate interpretation. For example, the ratio here means that after metric scores were calculated, the values for unprivileged groups (female) were divided by values for the privileged subgroup (male). In this example, except for the predictive equality ratio, the other measures are ε-non-discriminatory.

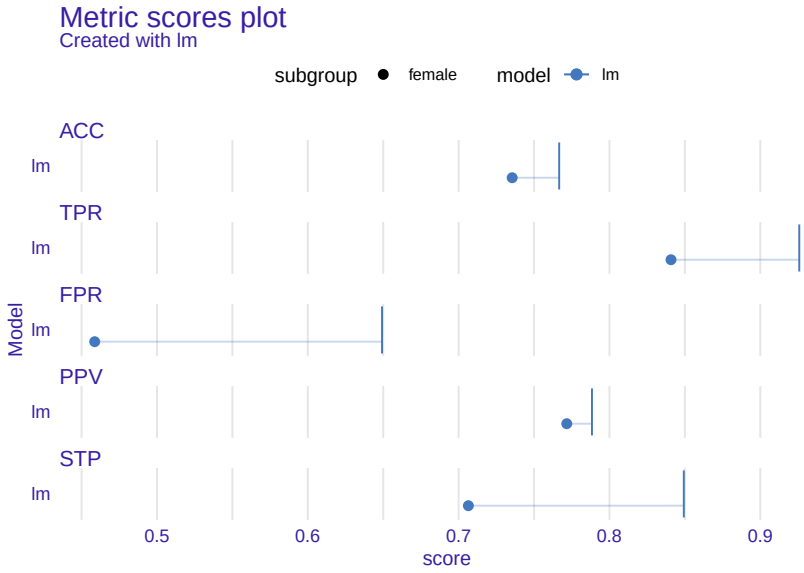


Figure 3: The Metric Scores plot summarises raw fairness metrics scores for subgroups. The dots stand for unprivileged subgroups (female) while vertical lines stans for the privileged subgroup (male). The horizontal lines act as a visual aid for measuring the difference between the scores of the metrics between the privileged and unprivileged subgroups.

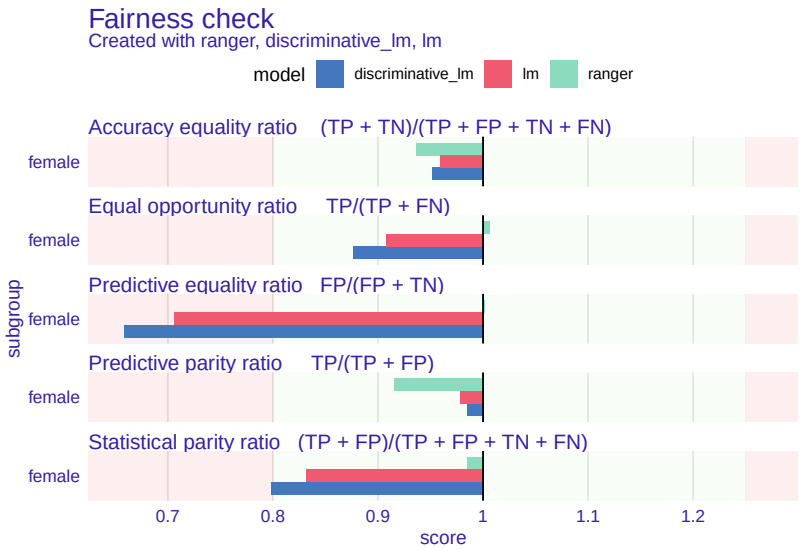


Figure 4: The Fairness Check plot for multiple models. It helps to compare models based on five selected fairness measures.

```
plot(metric_scores(fobject))
```

It is rare that a model perfectly meets all the fairness criteria. Therefore, a handy feature is the ability to compare several models on the same scale. We add two more explainers to the fairness assessment in the example below. Now `fairness_object` (in code: `fobject`) wraps three models together with different labels and cutoffs for subgroups. The `fairness_object` can be later used as a basis for another `fairness_object`. In detail, while running `fairness_check()` for the first time, explainer/explainers have to be provided along with three arguments described at the start of this section. However, as shown below, when providing explainers with a `fairness_object`, those arguments are not necessary as they are already a part of the previously created `fairness_object`.

First, let us create two more models based on the *German Credit Data*. The first will be a logistic regression model that uses fewer columns and has access to the `Sex` feature. The second is random forest from **ranger** (Wright and Ziegler, 2017). It will be trained on the whole dataset.

```
discriminative_lm_model <- glm(Risk~.,
         data   = german[c("Risk", "Sex","Age",
                "Checking.account", "Credit.amount")],
         family = binomial(link = "logit"))

library("ranger")
rf_model <- ranger::ranger(Risk ~.,
         data = german, probability = TRUE,
         max.depth = 4, seed = 123)
```

These models differ in the way how the predict function works. To unify operations on these models, we need to create **DALEX** explainer objects. The `label` argument specifies how these models are named on plots.

```
explainer_dlm <- DALEX::explain(discriminative_lm_model,
        data = german[c("Sex", "Age", "Checking.account", "Credit.amount")],
        y = y_numeric,
        label = "discriminative_lm")

explainer_rf <- DALEX::explain(rf_model,
        data = german[,-1], y = y_numeric)
```

Now we are ready to assess fairness. The resulting plot is presented in Figure 4.

```
fobject <- fairness_check(explainer_rf, explainer_dlm, fobject)
plot(fobject)
```

When plotted, new bars appear on the fairness check plot. Those are new metric scores for added models. This information can be summarized in a numerical way with the `print()` function.

```
print(fobject, colorize = FALSE)

#>
#> Fairness check for models: ranger, discriminative_lm, lm
#>
#> ranger passes 5/5 metrics
#> Total loss :  0.1699186
#>
#> discriminative_lm passes 3/5 metrics
#> Total loss :  0.7294678
#>
#> lm passes 4/5 metrics
#> Total loss :  0.6153324
```

3 Package architecture

The **fairmodels** package provides a unified interface for predictive models independently of their internal structure. Using a model agnostic approach with **DALEX** explainers facilitates this process (Biecek, 2018). There is a unified way for each explainer to check if explained model lives up to user fairness standards. Checking fairness with **fairmodels** is straightforward and can be done with the three-step pipeline.

```
classification model  |>  explain()  |>  fairness_check()
```

The output of such a pipeline is an object of class `fairness_object`, a unified structure to wrap model explainer or multiple model explainers and other `fairness_objects` in a single container. Aggregation of fairness measures is done based on groups defined by model labels. This is why model explainers (even those wrapped by `fairness_objects`) must have different labels. Moreover, some visualizations for model comparison assume that all models are created from the same data. Of course, each model can use different variables or different feature transformations, but the order and number of rows shall stay the same. To facilitate aggregation of models **fairmodels** allows creating `fairness_objects` in other ways:

- `explainers |> fairness_check()` - possibly many explainers can be passed to `fairness_check()`,
- `fairness_objects |> fairness_check()` - explainers stored in `fairness_objects` passed to `fairness_check()` will be aggregated into one `fairness_object`,
- `explainer` & `fairness_objects |> fairness_check()` - explainers passed directly and explainers from `fairness_objects` will be aggregated into one `fairness_object`.

When using the last two pipelines, protected vectors and privileged parameters are assumed to be the same, so passing them to `fairness_check()` is unnecessary.

To create a `fairness_object`, at least one explainer needs to be passed to `fairness_check()` function, which returns the said object. `fairness_object` metrics for each subgroup are calculated from the separate confusion matrices.

The `fairness_object` has numerous fields. Some of them are:

- `parity_loss_metric_data` - data.frame containing parity loss for each metric and classifier,
- `groups_data` - list of metric scores for each metric and model,
- `group_confusion_matrices` - list of values in confusion matrices for each model and metric,
- `explainers` - list of DALEX explainers. When explainers and/or `fairness_object` are added, then explainers and/or explainers extracted from `fairness_object` are added to that list,
- `label` - character vector of labels for each explainer.
- `...` - other fields.

The `fairness_object` methods are used to create numerous objects that help to visualize bias. In the next sections, we list more detailed functions for deeper exploration of bias. Detailed relations between objects created with **fairmodels** are depicted in Figure 5. The general overview of the workflow is presented in Figure 6.

4 Visualizing bias

In **fairmodels** there are 12 metrics based on confusion matrices for each subgroup, see the following table for the complete list. Some of them were already introduced before.

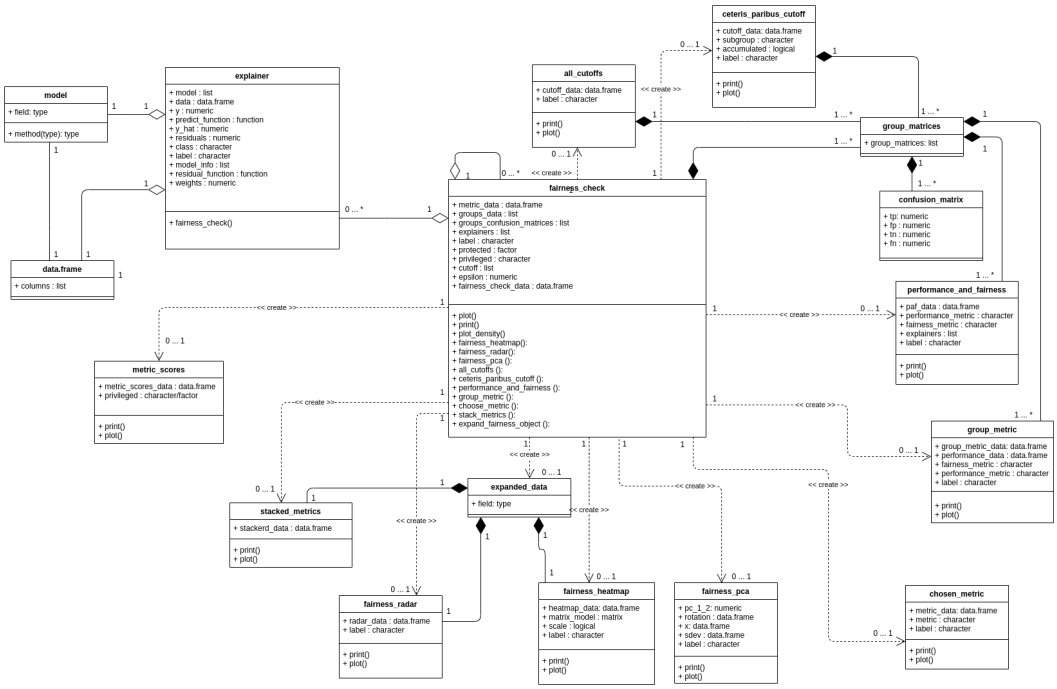


Figure 5: Class diagram for objects created by functions from the fairmodels package. Each rectangle corresponds to one class, the name of this class is in the header of the rectangle. Each of these classes is a list containing a certain list of objects. The top slot lists the names and types of each object the list. The bottom slot contains a list of functions that can be performed on objects of the specified class. If two classes are connected by a line ending in a diamond it means that one class contains objects of the other class. If two rectangles are connected by a dashed line, it means that on the basis of one object, an object of another class can be produced. In this case, more detailed fairness statistics can be produced from the central object of the fairness check class. See the full resolution at https://bit.ly/3HNbNvo

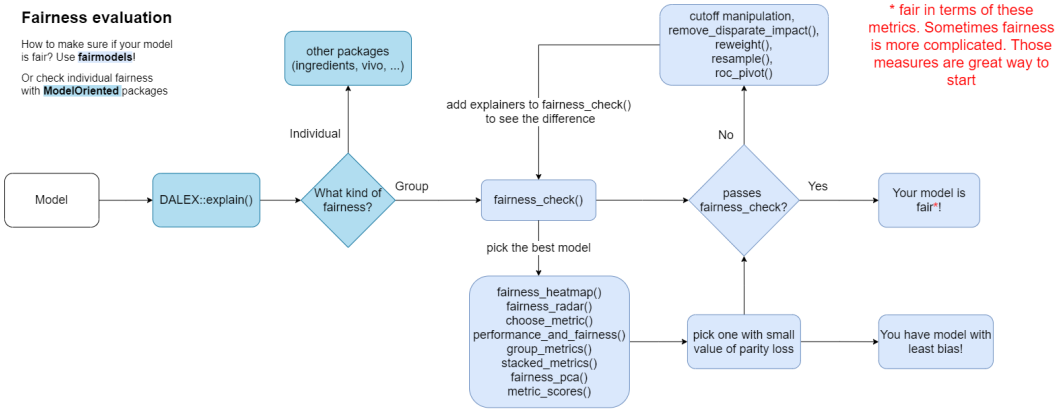


Figure 6: Flowchart for the fairness assessment with the fairmodels package. The arrows describe typical sequences of actions when exploring the fairness of the models. For ease of use, the names of the functions that can be used in a given step are indicated. Note that this procedure is intended to look at the model from multiple perspectives in order to track down potential problems in the model. Merely satisfying the fairness criteria does not automatically mean that the model is free of any errors

Metric	Formula	Name	Fairness criteria
TPR	$\frac{TP}{TP+FN}$	True positive rate	Equal opportunity (Hardt et al., 2016)
TNR	$\frac{TN}{TN+FP}$	True negative rate	
PPV	$\frac{TP}{TP+FP}$	Positive predictive value	Predictive parity (Chouldechova, 2016)
NPV	$\frac{TN}{TN+FN}$	Negative predictive value	
FNR	$\frac{FN}{FN+TP}$	False negative rate	
FPR	$\frac{FP}{FP+TN}$	False positive rate	Predictive equality (Corbett-Davies et al., 2017)
FDR	$\frac{FP}{FP+TP}$	False discovery rate	
FOR	$\frac{FN}{FN+TN}$	False omission rate	
TS	$\frac{TP}{TP+FN+FP}$	Threat score	
STP	$\frac{TP+FP}{TP+FP+TN+FN}$	Positive rate	Statistical parity (Dwork et al., 2012)
ACC	$\frac{TP+TN}{TP+TN+FP+FN}$	Accuracy	Overall accuracy equality (Berk et al., 2017)
F1	$\frac{2 \cdot PPV * TPR}{PPV+TPR}$	F1 score	

Table 1: Fairness metrics implemented in the **fairmodels** package

Not all metrics are needed to determine if the discrimination exists, but they are helpful to acquire a fuller picture. To facilitate the visualization over many subgroups, we introduce a function that maps metric scores among subgroups to a single value. This function, which we call `parity_loss`, has an attractive property. Due to the usage of the absolute value of the natural logarithm, it will return the same value whether the ratio is inverted or not.

So, for example, when we would like to know the parity loss of Statistical Parity between unprivileged (b) and privileged (a) subgroups, we mean value like this:

$$STP_{parity\ loss} = \left| \ln \left(\frac{STP_b}{STP_a} \right) \right|. \tag{2}$$

This notation is very helpful because it allows to accumulate $STP_{parity\ loss}$ overall unprivileged subgroups, so not only in the binary case.

$$STP_{parity\ loss} = \sum_{i \in \{a,b,\ldots\}} \left| \ln \left(\frac{STP_i}{STP_a} \right) \right|. \tag{3}$$

The `parity_loss` relates strictly to ratios. The classifier is more fair if `parity_loss` is low. This property is helpful in visualizations.

There are several modifying functions that operate on `fairness_object`. Their usage will return other objects. The relations between them is depicted on the class diagram (Figure 5). The objects can then be plotted with a generic `plot()` function. Additionally, a special plotting function works immediately on `fairness_object`, which is `plot_density`. The user can directly specify which metrics shall be visible in the plot in some functions. The detailed technical introduction for all these functions is presented in **fairmodels** manual.

Plots visualizing different aspects of `parity_loss` can be created with one of the following pipelines:

- `fairness_object |> modifying_function(...) |> plot()`
  This pipe is preferred and allows setting parameters in both modifying functions and certain plot functions, which is not the case with the next pipeline.
- `fairness_object |> plot_fairmodels(type = modifying_function, ...)`
  Additional parameters are passed to the modifying functions and not to the plot function.

Using the pipelines, different plots can be obtained by superseding the `modifying_function` with function names. Four examples of additional graphical functions available in the `fairmodels` can be seen in Figure 7. This package implements a total of 8 different diagnostic plots, each describing a different fairness perspective. To see different aspects of fairness and bias, the user can choose the model with the smallest bias, find out the similarity between metrics and models, compare models in both fairness and performance, and see how cutoff manipulation might change the `parity_loss`. Find more information about each of them in the documentation.

```
fp1  <- plot(ceteris_paribus_cutoff(fobject, "male", cumulated=TRUE))
fp2  <- plot(fairness_heatmap(fobject))
fp3  <- plot(stack_metrics(fobject))
fp4  <- plot(plot_density(fobject))

library("patchwork")
fp1 + fp2 + fp3 + fp4 +
  plot_layout(ncol = 2)
```

5 Bias mitigation

What can be done if the model does not meet the fairness criteria? Machine learning practitioners might use other algorithms or variables to construct unbiased models, but this does not guarantee passing the `fairness_check()`. An alternative is to use bias mitigation techniques that adjust the data or model to meet fairness conditions. There are essentially three types of such methods. The first is data pre-processing. There are many ways to "correct" the data when there are unwanted correlations between variables or sample sizes among subgroups in data. The second one is in-processing, which is, for example, optimizing classifiers not only to reduce classification error but also to minimize a fairness metric. Last but not least is post-processing which modifies model output so that predictions and miss-predictions among subgroups are more alike.

The **fairmodels** package offers five functions for bias mitigation, three for pre-processing, and two for post-processing. Most of these approaches are also implemented in aif360 (Bellamy et al., 2018). However, in **fairmodels** there are separate implementations of them in R. There are a lot of useful mitigation techniques that are not in **fairmodels** like those in Hardt et al. (2016) and numerous in-processing algorithms.

Data pre-processing

- **Disparate impact remover**
  In **fairmodels** geometric repair, an algorithm originally introduced by Feldman et al. (2015), works on ordinal, numeric features. Depending on the $\lambda \in [0, 1]$ parameter, this method will transform the distribution of a given feature. The idea is simple. Given feature distribution in different subgroups, the algorithm finds optimal distribution (according to earth mover's distance) and transforms distribution for each subgroup to match the optimal one. For example, if age is an important feature and its distribution is different in two subgroups, and we want to change that, then the geometric repair will map each individual's age to a new distribution (different age). It will be preserving the order - the ranks (in our case, seniority) of observations are preserved. Parameter λ is responsible for the repair degree, so for full repair, lambda should be set to 1. The method does not focus on a particular metric but rather tries to level out them by transforming potentially harmful feature distributions.
- **Reweighting**
  Reweighting is a rather straightforward approach. This method was implemented according to Kamiran and Calders (2011). It computes weights by dividing the theoretical probability of assigning favorable labels for a subgroup by real (observed) probability (based on the data). Theoretic probability for a subgroup is computed by multiplying the probability of assigning a

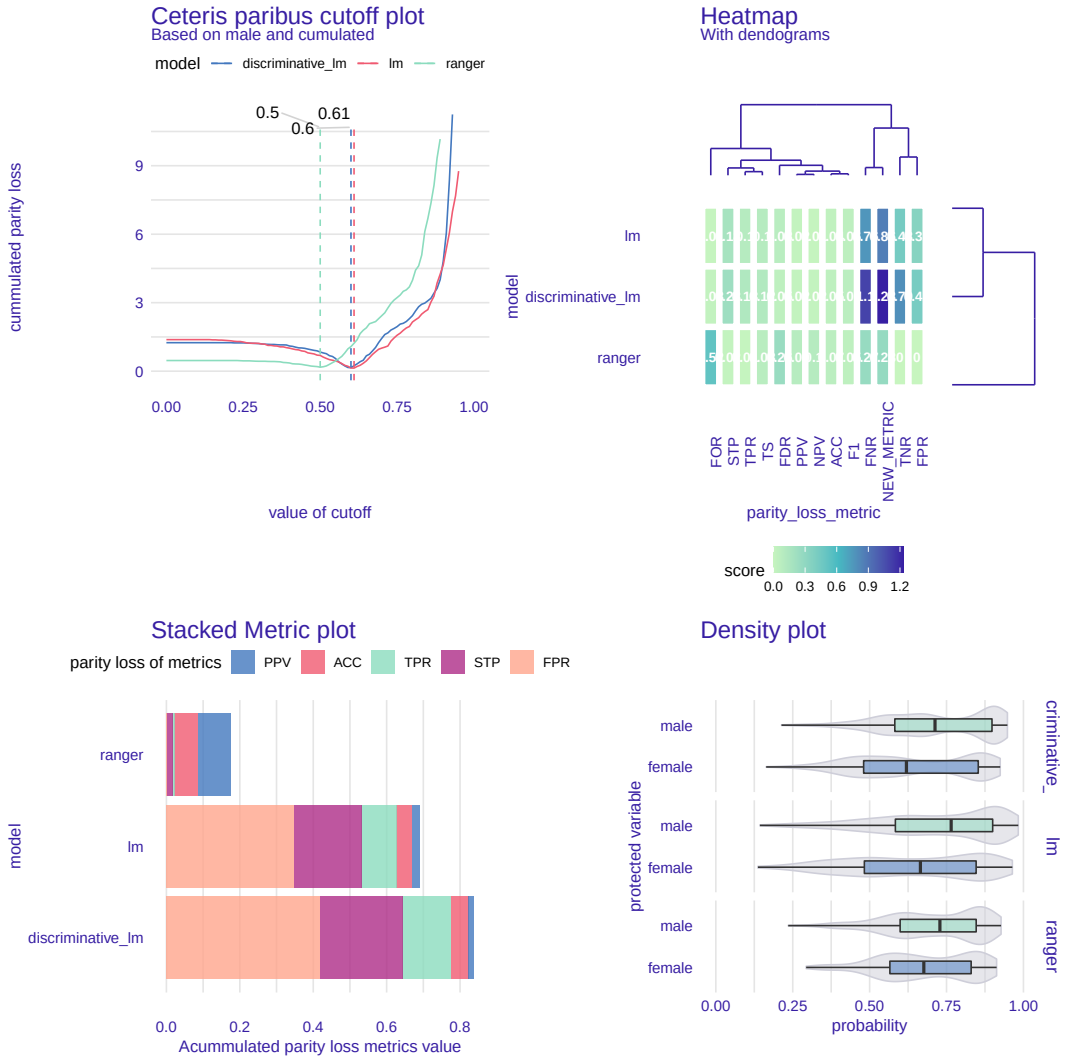


Figure 7: Four examples of additional graphical functions are available in the fairmodels package that facilitates model and bias exploration. The Ceteris Paribus Cuttoff plot helps select the cutoff values for each model to maximize a particular measure of fairness. In this case, the suggested cutoff point for both linear models is similar. However, the ranger model does not have calibrated probabilities and thus requires a different cutoff. The Heatmap plot is very helpful when comparing large numbers of models. It shows profiles of selected fairness measures for each of the models under consideration. In this case, the fairness profiles for both linear models are similar. The Stacked Metric plot helps you compare models by summing five different fairness measures. The different layers of this plot allow you to compare individual measures, but if you don't know which one to focus on, it is useful to look at the sum of the measures. In this case, the ranger model has the highest fairness values. Finally, the Density plot helps to compare the score distributions of the models between the advantaged and disadvantaged groups. In this case, we find that for females the distributions of the scores are lower in all models, with the largest difference for the lm model.

favorable label (for all populations) by picking observation from a certain subgroup. It focuses on mitigating statistical parity.

- **Resampling**
  Resampling is based on weights calculated in `reweighting`. Each weight for a subgroup is multiplied by the size of the subgroup. Then, whether the subgroup is deprived or not (if weight is higher than one, the subgroup is considered deprived), observations are duplicated from either one that were assigned a favorable label or not. There are two types of resampling-uniform and preferential. The uniform is making algorithm pick or omit observations randomly without considering its probabilistic score. Preferential uses another probabilistic classifier, potentially different from the main model for final predictions. In Kamiran and Calders (2011) it is called ranker - it predicts the probabilities for the observations to decide which observations are close to the cutoff border (usually 0.5). Based on the probabilistic output of the ranker, the observations are sorted, and the ones with the highest/lowest ranks are either left out or duplicated depending on the case—more on that on Kamiran and Calders (2011). The **fairmodels** implementation, instead of training the ranker as in the aforementioned paper, uses a vector of previously calculated probabilities provided by the user. With this, it shifts the decision and responsibility of choosing a ranker to the user. It focuses on mitigating statistical parity.

Model post-processing

- **Reject Option based Classification Pivot**
  The `roc_pivot` method is implemented based on Kamiran et al. (2012) in the **fairmodels** package. Let $\theta \in (0, 1)$ be the value that determines the radius of the so-called critical region, which is an area around the cutoff. The user specifies the θ, and it should describe how big the critical region should be. For example if $\theta = 0.1$ and cutoff is 0.6, then the critical region will be (0.5, 0.7). Let's assume that we are predicting a favorable outcome. If the assigned probability of observation is in the described region, then the probabilities are pivoting on the other side of the cutoff with a certain assumption. If an observation in a critical region is considered to be the privileged and it is on the right side of the cutoff, then its probabilities are pivoting from the right side of the cutoff to the left. So if an observation is in the critical region and it is considered unprivileged, then if it is on the left side of the cutoff, it will pivot to the right side. Pivoting here means changing the side of the cutoff so that the distance from the cutoff stays unchanged. It does not intend to mitigate a single metric but rather changes predictions in the critical region (the region with low certainty). By pivoting the predictions, it might lower more metrics.

- **Cutoff manipulation**
  The **fairmodels** package supports setting cutoff for each subgroup. Users may pick `parity_loss` metrics of their choice and find the minimal `parity_loss`. It is part of `ceteris_paribus_cutoff()` function. Based on picked metrics, the sum of parity loss is calculated for each cutoff of the chosen subgroup. Then the minimal value is found—this way, optimal values might be found for metrics of interest. The minimum is marked with a dashed vertical line (see Figure 7). This approach however might be to some extent concerning. Some might argue that setting different cutoffs for different subgroups is unfair and is punishing privileged subgroups for something they have no control of. Especially in the individual fairness field, it would be concerning if two similar people with different sensitive attributes would have two different thresholds and potentially two different outcomes. This is a valid point, and this method should be used with knowledge of all its drawbacks. The cutoff manipulation method targets metrics chosen by the user.

All pre-processing methods can be used with two pipelines, whereas post-processing can be used in one specific way.

- Pre-processing pipelines
  - `data/explainer |> method`
    Returns either weights, indexes, or changed data depending on the method used.
  - `data/explainer |> pre_process_data(data, protected, y, type = ...)`
    Always returns data.frame. In case of weights data has additional column called `_weights_`.
- Post-processing pipelines
  - `fairness_object |> ceteris_paribus_cutoff(subgroup, ...) |> print/plot`
    This is the pipeline for creating ceteris paribus cutoff print and plot.
  - `explainer |> roc_pivot(protected, privileged, ...)`
    The pipeline will return explainer with `y_hat` field changed.

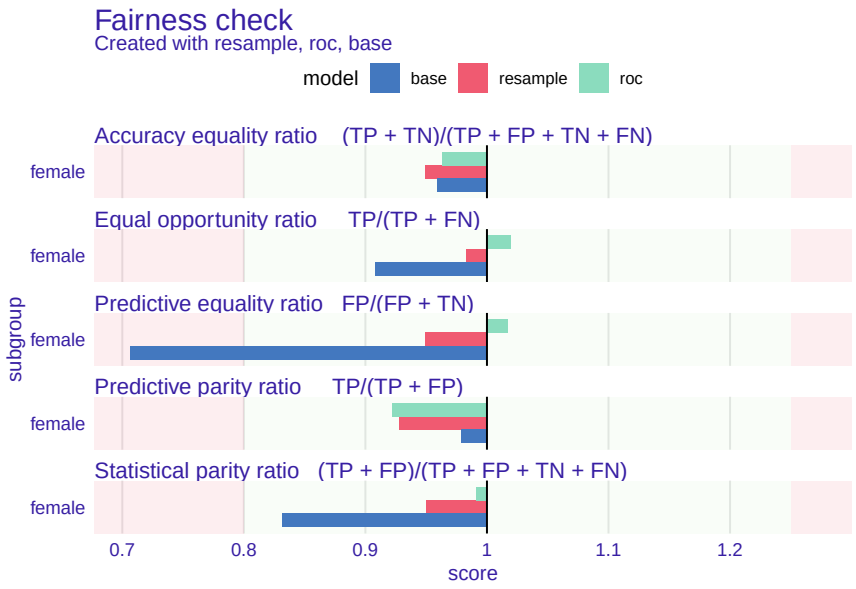


Figure 8: Graphical summary of a base model (blue bars) and model after applying two bias mitigation techniques (red and green bars). By comparing adjacent rectangles one can read how the respective technique affected the corresponding fairness measure

The user should be aware that debiasing one metric might enhance bias in another. It is a so-called fairness-fairness trade-off. There is also a fairness-performance trade-off where debiasing one metric leads to worse performance. Another thing to remember is, as found in Agrawal et al. (2020), metrics might not generalize well to out-of-distribution examples, so it is advised also to check the fairness metrics on a separate test set.

Example

Now we will show an example usage of one pre-processing and one post-processing method. As before, the **German Credit Data** will be used along with the previously created `lm_model`. So firstly, we create a new dataset using `pre_process_data` and then we use it to train the logistic regression classifier.

```
resampled_german   <- german |> pre_process_data(protected = german$Sex,
                y_numeric, type = 'resample_uniform')

lm_model_resample  <- glm(Risk~.,
                data   = resampled_german,
                family = binomial(link = "logit"))

explainer_lm_resample <- DALEX::explain(lm_model_resample,
                data = german[,-1], y = y_numeric, verbose = FALSE)
```

Then we make other explainers. We use previously created `explainer_lm` with the post-processing function `roc_pivot`. We set parameter `theta = 0.05` for a rather narrow area of a pivot.

```
new_explainer <- explainer_lm |> roc_pivot(protected = german$Sex,
                privileged = "male", theta = 0.05)
```

In the end, we create `fairness_object` with explainers obtained with the code above and one created in the first example to see the difference.

```
fobject <- fairness_check(explainer_lm_resample, new_explainer, explainer_lm,
                protected = german$Sex, privileged = "male",
                label = c("resample", "roc", "base"),
                verbose = FALSE)

fobject |> plot()
```

The result of the code above is presented in Figure 8. The mitigation methods successfully eliminated bias in all of the metrics. Both models are better than the original base. This is not always the case - sometimes, eliminating bias in one metric may increase bias in another metric. For example, let's consider a perfectly accurate model, but some subgroups receive few positive predictions (bias in Statistical parity). In that case, mitigating the bias in Statistical parity would decrease the Accuracy equality ratio.

6 Summary and future work

This paper showed that checking for bias in machine learning models can be done conveniently and flexibly. The package **fairmodels** described above is a self-sufficient tool for bias detection, visualization, and mitigation in classification machine learning models. We presented theory, package architecture, suggested usage, and examples along with plots. Along the way, we introduced the core concepts and assumptions that come along the bias detection and plot interpretation. The package is still improved and enhanced, which can be seen by adding the announced regression module based on Steinberg et al. (2020). We did not cover it in this article because it is still an experimental tool. Another tool for in-processing classification closely related to **fairmodels** has also been added and can be found on https://github.com/ModelOriented/FairPAN.

The source code of the package, vignettes, examples, and documentation can be found at https://modeloriented.github.io/fairmodels/. The stable version is available on CRAN. The code and the development version can be found on GitHub https://github.com/ModelOriented/fairmodels. This is also a place to report bugs or requests (through GitHub issues).

In the future, we plan to enhance the spectrum of bias visualization plots and introduce regression and individual fairness methods. The potential way to explore would be an in-processing bias mitigation - training models that minimize cost function and adhere to certain fairness criteria. This field is heavily developed in Python and lacks appropriate attention in R.

Acknowledgements

Work on this package was financially supported by the NCN Sonata Bis-9 grant 2019/34/E/ST6/00052.

Bibliography

A. Agrawal, F. Pfisterer, B. Bischl, J. Chen, S. Sood, S. Shah, F. Buet-Golfouse, B. A. Mateen, and S. Vollmer. Debiasing classifiers: Is reality at variance with expectation? *Electronic*, 2020. [p240]

J. Angwin, J. Larson, S. Mattu, , and L. Kirchner. Machine bias: There's software used across the country to predict future criminals. And it's biased against blacks. *ProPublica*, 2016. URL https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. [p227]

S. Barocas, M. Hardt, and A. Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org. [p227, 228, 229, 230]

R. K. E. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, S. Nagar, K. N. Ramamurthy, J. Richards, D. Saha, P. Sattigeri, M. Singh, K. R. Varshney, and Y. Zhang. AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias, Oct. 2018. URL https://arxiv.org/abs/1810.01943. [p228, 237]

R. Berk, H. Heidari, S. Jabbari, M. Kearns, and A. Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, 03 2017. URL https://doi.org/10.1177/0049124118782533. [p229, 236]

P. Biecek. DALEX: Explainers for Complex Predictive Models in R. *Journal of Machine Learning Research*, 19(84):1–5, 2018. URL http://jmlr.org/papers/v19/18-416.html. [p231, 234]

P. Biecek and T. Burzykowski. *Explanatory Model Analysis*. Chapman and Hall/CRC, New York, 2021. ISBN 9780367135591. URL https://pbiecek.github.io/ema/. [p228]

R. Binns. On the apparent conflict between individual and group fairness. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 514–524, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369367. doi: 10.1145/3351095.3372864. URL https://doi.org/10.1145/3351095.3372864. [p228]

S. Bird, M. Dudík, R. Edgar, B. Horn, R. Lutz, V. Milan, M. Sameki, H. Wallach, and K. Walker. Fairlearn: A toolkit for assessing and improving fairness in AI. Technical Report MSR-TR-2020-32, Microsoft, 2020. URL https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai/. [p228]

J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In S. A. Friedler and C. Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA, 23–24 Feb 2018. URL http://proceedings.mlr.press/v81/buolamwini18a.html. [p227]

A. Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, 5, 10 2016. URL https://doi.org/10.1089/big.2016.0047. [p229, 236]

D. Cirillo, S. Catuara-Solarz, C. Morey, E. Guney, L. Subirats, S. Mellino, A. Gigante, A. Valencia, M. J. Rementeria, A. S. Chadha, and N. Mavridis. Sex and gender differences and biases in artificial intelligence for biomedicine and healthcare. *npj Digital Medicine*, 3(1):81, 2020. ISSN 2398-6352. doi: 10.1038/s41746-020-0288-5. URL https://doi.org/10.1038/s41746-020-0288-5. [p228]

Code of Federal Regulations. Section 4d, uniform guidelines on employee selection procedures (1978), 1978. URL https://www.govinfo.gov/content/pkg/CFR-2014-title29-vol4/xml/CFR-2014-title29-vol4-part1607.xml. [p230]

S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, and A. Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 797–806, New York, NY, USA, 2017. Association for Computing Machinery. URL https://doi.org/10.1145/3097983.3098095. [p229, 236]

Council of Europe. Guidelines on facial recognition, 28 Jan 2021. URL https://www.coe.int/en/web/portal/-/facial-recognition-strict-regulation-is-needed-to-prevent-human-rights-violations-. [p228]

D. Dua and C. Graff. UCI machine learning repository, 2017. URL https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data). [p231]

C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 214–226, New York, NY, USA, 2012. Association for Computing Machinery. URL https://doi.org/10.1145/2090236.2090255. [p228, 229, 236]

European Union Agency for Fundamental Rights. Handbook on european non-discrimination law, 2018. [p230]

European Union Agency for Fundamental Rights and Council of Europe. *Handbook on European non-discrimination law*. Luxembourg: Publications Office of the European Union, 2018. https://fra.europa.eu/en/publication/2018/handbook-european-non-discrimination-law-2018-edition. [p227]

M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 259–268, New York, NY, USA, 2015. Association for Computing Machinery. URL https://doi.org/10.1145/2783258.2783311. [p237]

H2O.ai. *H2O AutoML*, June 2017. URL http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html. H2O version 3.30.0.1. [p228]

M. Hardt, E. Price, E. Price, and N. Srebro. Equality of opportunity in supervised learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3315–3323. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6374-equality-of-opportunity-in-supervised-learning.pdf. [p229, 230, 236, 237]

F. Kamiran and T. Calders. Data pre-processing techniques for classification without discrimination. *Knowledge and Information Systems*, 33, 10 2011. URL https://doi.org/10.1007/s10115-011-0463-8. [p227, 237, 239]

F. Kamiran, A. Karim, and X. Zhang. Decision theory for discrimination-aware classification. In *2012 IEEE 12th International Conference on Data Mining*, pages 924–929, 2012. URL https://doi.org/10.1109/ICDM.2012.45. [p239]

N. Kozodoi and T. V. Varga. *fairness: Algorithmic Fairness Metrics*, 2021. URL https://CRAN.R-project.org/package=fairness. R package version 1.2.1. [p228]

N. Kozodoi, J. Jacob, and S. Lessmann. Fairness in credit scoring: Assessment, implementation and profit implications. *European Journal of Operational Research*, 2021. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.06.023. URL https://doi.org/10.1016/j.ejor.2021.06.023. [p230]

P. Lahoti, K. P. Gummadi, and G. Weikum. [ifair: Learning individually fair data representations for algorithmic decision making]. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1334–1345, 2019. URL https://doi.org/10.1109/ICDE.2019.00121. [p227]

N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning, 2019. URL https://arxiv.org/abs/1908.09635. [p227]

D. Plečko and N. Meinshausen. Fair data adaptation with quantile preservation, 2019. URL https://arxiv.org/abs/1911.06685. [p228]

P. Saleiro, B. Kuester, A. Stevens, A. Anisfeld, L. Hinkson, J. London, and R. Ghani. Aequitas: A bias and fairness audit toolkit, 2018. URL https://arxiv.org/abs/1811.05577. [p228]

D. C. Steinberg, A. Reid, and S. T. O'Callaghan. Fairness measures for regression via probabilistic classification. *ArXiv*, abs/2001.06089, 2020. [p241]

M. Wick, S. Panda, and J.-B. Tristan. Unlocking fairness: a trade-off revisited. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8783–8792. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/373e4c5d8edfa8b74fd4b6791d0cf6dc-Paper.pdf. [p227]

M. N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017. doi: 10.18637/jss.v077.i01. [p233]

Jakub Wiśniewski
Warsaw University of Technology
Faculty of Mathematics and Information Science
Poland
jakwisn@gmail.com

Przemysław Biecek
Warsaw University of Technology
Faculty of Mathematics and Information Science
University of Warsaw
Faculty of Mathematics, Informatics, and Mechanics
Poland
https://pbiecek.github.io/
ORCiD: 0000-0001-8423-1823
przemyslaw.biecek@pw.edu.pl

Palmer Archipelago Penguins Data in the palmerpenguins R Package - An Alternative to Anderson's Irises

by Allison M. Horst, Alison Presmanes Hill, and Kristen B. Gorman

Abstract In 1935, Edgar Anderson collected size measurements for 150 flowers from three species of *Iris* on the Gaspé Peninsula in Quebec, Canada. Since then, Anderson's *Iris* observations have become a classic dataset in statistics, machine learning, and data science teaching materials. It is included in the base R **datasets** package as `iris`, making it easy for users to access without knowing much about it. However, the lack of data documentation, presence of non-intuitive variables (e.g. "sepal width"), and perfectly balanced groups with zero missing values make `iris` an inadequate and stale dataset for teaching and learning modern data science skills. Users would benefit from working with a more representative, real-world environmental dataset with a clear link to current scientific research. Importantly, Anderson's *Iris* data appeared in a 1936 publication by R. A. Fisher in the *Annals of Eugenics* (which is often the first-listed citation for the dataset), inextricably linking `iris` to eugenics research. Thus, a modern alternative to `iris` is needed. In this paper, we introduce the **palmerpenguins** R package (Horst et al., 2020), which includes body size measurements collected from 2007 - 2009 for three species of *Pygoscelis* penguins that breed on islands throughout the Palmer Archipelago, Antarctica. The `penguins` dataset in **palmerpenguins** provides an approachable, charismatic, and near drop-in replacement for `iris` with topical relevance for polar climate change and environmental impacts on marine predators. Since the release on CRAN in July 2020, the **palmerpenguins** package has been downloaded over 462,000 times, highlighting the demand and widespread adoption of this viable `iris` alternative. We directly compare the `iris` and `penguins` datasets for selected analyses to demonstrate that R users, in particular teachers and learners currently using `iris`, can switch to the Palmer Archipelago penguins for many use cases including data wrangling, visualization, linear modeling, multivariate analysis (e.g., PCA), cluster analysis and classification (e.g., by k-means).

Introduction

In 1935, American botanist Edgar Anderson measured petal and sepal structural dimensions (length and width) for 50 flowers from three *Iris* species: *Iris setosa*, *Iris versicolor*, and *Iris virginica* (Anderson, 1935). The manageable but non-trivial size (5 variables and 150 total observations) and characteristics of Anderson's *Iris* dataset, including linear relationships and multivariate normality, have made it amenable for introducing a wide range of statistical methods including data wrangling, visualization, linear modeling, multivariate analyses, and machine learning. The *Iris* dataset is built into a number of software packages including the auto-installed datasets package in R (as `iris`, R Core Team, 2021), Python's scikit-learn machine learning library (Pedregosa et al., 2011), and the SAS Sashelp library (SAS Institute, Cary NC), which has facilitated its widespread use. As a result, eighty-six years after the data were initially published, the *Iris* dataset remains ubiquitous in statistics, computational methods, software documentation, and data science courses and materials.

There are a number of reasons that modern data science practitioners and educators may want to move on from `iris`. First, the dataset lacks metadata (Anderson, 1935), which does not reinforce best practices and limits meaningful interpretation and discussion of research methods, analyses, and outcomes. Of the five variables in `iris`, two (`Sepal.Width` and `Sepal.Length`) are not intuitive for most non-botanists. Even with explanation, the difference between *petal* and *sepal* dimensions is not obvious. Second, `iris` contains equal sample sizes for each of the three species ($n = 50$) with no missing values, which is cleaner than most real-world data that learners are likely to encounter. Third, the single factor (`Species`) in `iris` limits options for analyses. Finally, due to its publication in the *Annals of Eugenics* by statistician R.A. Fisher (Fisher, 1936), `iris` is burdened by a history in eugenics research, which we are committed to addressing through the development of new data science education products as described below.

Given the growing need for fresh data science-ready datasets, we sought to identify an alternative dataset that could be made easily accessible for a broad audience. After evaluating the positive and negative features of `iris` in data science and statistics materials, we established the following criteria for a suitable alternative:

- Available by appropriate license like a Creative Commons 0 license (CC0 "no rights reserved")
- Feature intuitive subjects and variables that are interesting and understandable to learners across disciplines

- Complete metadata and documentation
- Manageable (but not trivial) in size
- Minimal data cleaning and pre-processing required for most analyses
- Real-world (not manufactured) modern data
- Provides similar opportunities for teaching and learning R, data science, and statistical skills
- Can easily replace `iris` for most use cases

Here, we describe an alternative to `iris` that largely satisfies these criteria: a refreshing, approachable, and charismatic dataset containing real-world body size measurements for three *Pygoscelis* penguin species that breed throughout the Western Antarctic Peninsula region, made available through the United States Long-Term Ecological Research (US LTER) Network. By comparing data structure, size, and a range of analyses side-by-side for the two datasets, we demonstrate that the Palmer Archipelago penguin data are an ideal substitute for `iris` for many use cases in statistics and data science education.

Data source

Body size measurements (bill length and depth, flipper length - flippers are the modified "wings" of penguins used for maneuvering in water, and body mass), clutch (i.e., egg laying) observations (e.g., date of first egg laid, and clutch completion), and carbon ($^{13}C/^{12}C$, $\delta^{13}C$) and nitrogen ($^{15}N/^{14}N$, $\delta^{15}N$) stable isotope values of red blood cells for adult male and female Adélie (*P. adeliae*), chinstrap (*P. antarcticus*), and gentoo (*P. papua*) penguins on three islands (Biscoe, Dream, and Torgersen) within the Palmer Archipelago were collected from 2007 - 2009 by Dr. Kristen Gorman in collaboration with the Palmer Station LTER, part of the US LTER Network. For complete data collection methods and published analyses, see Gorman et al. (2014). Throughout this paper, penguins species are referred to as "Adélie", "Chinstrap", and "Gentoo".

The data in the **palmerpenguins** R package are available for use by CC0 license ("No Rights Reserved") in accordance with the Palmer Station LTER Data Policy and the LTER Data Access Policy, and were imported from the Environmental Data Initiative (EDI) Data Portal at the links below:

- Adélie penguin data (Palmer Station Antarctica LTER and Gorman, 2020a): KNB-LTER Data Package 219.5
- Gentoo penguin data (Palmer Station Antarctica LTER and Gorman, 2020b): KNB-LTER Data Package 220.5
- Chinstrap penguin data (Palmer Station Antarctica LTER and Gorman, 2020c): KNB-LTER Data Package 221.6

The palmerpenguins R package

R users can install the **palmerpenguins** package from CRAN:

```
install.packages("palmerpenguins")
```

Information, examples, and links to community-contributed materials are available on the **palmerpenguins** package website: allisonhorst.github.io/palmerpenguins/. See the Appendix for how Python and Julia users can access the same data.

The **palmerpenguins** R package contains two data objects: `penguins_raw` and `penguins`. The `penguins_raw` data consists of all raw data for 17 variables, recorded completely or in part for 344 individual penguins, accessed directly from EDI (`penguins_raw` properties are summarized in Appendix B). We generally recommend using the curated data in `penguins`, which is a subset of `penguins_raw` retaining all 344 observations, minimally updated (Appendix A) and reduced to the following eight variables:

- `species`: a factor denoting the penguin species (Adélie, Chinstrap, or Gentoo)
- `island`: a factor denoting the Palmer Archipelago island in Antarctica where each penguin was observed (Biscoe Point, Dream Island, or Torgersen Island)
- `bill_length_mm`: a number denoting length of the dorsal ridge of a penguin bill (millimeters)
- `bill_depth_mm`: a number denoting the depth of a penguin bill (millimeters)
- `flipper_length_mm`: an integer denoting the length of a penguin flipper (millimeters)
- `body_mass_g`: an integer denoting the weight of a penguin's body (grams)
- `sex`: a factor denoting the sex of a penguin sex (male, female) based on molecular data
- `year`: an integer denoting the year of study (2007, 2008, or 2009)

Table 1: Overview comparison of **penguins** and **iris** dataset features and characteristics.

Feature	iris	penguins
Year(s) collected	1935	2007 - 2009
Dimensions (col x row)	5 x 150	8 x 344
Documentation	minimal	complete metadata
Variable classes	double (4), factor (1)	double (2), int (3), factor (3)
Missing values?	no (n = 0; 0.0%)	yes (n = 19; 0.7%)

Table 2: Grouped sample size for **iris** (by species; n = 150 total) and **penguins** (by species and sex; n = 344 total). Data in **penguins** can be further grouped by island and study year.

iris sample size (by species)		penguins sample size (by species and sex)			
Iris species	Sample size	Penguin species	Female	Male	NA
setosa	50	Adélie	73	73	6
versicolor	50	Chinstrap	34	34	0
virginica	50	Gentoo	58	61	5

The same data exist as comma-separated value (CSV) files in the package ("penguins_raw.csv" and "penguins.csv"), and can be read in using the built-in `path_to_file()` function in **palmerpenguins**. For example,

```
library(palmerpenguins)
df <- read.csv(path_to_file("penguins.csv"))
```

will read in "penguins.csv" as if from an external file, thus automatically parsing `species`, `island`, and `sex` variables as characters instead of factors. This option allows users opportunities to practice or demonstrate reading in data from a CSV, then updating variable class (e.g., characters to factors).

Comparing `iris` and `penguins`

The `penguins` data in **palmerpenguins** is useful and approachable for data science and statistics education, and is uniquely well-suited to replace the `iris` dataset. Comparisons presented are selected examples for common `iris` uses, and are not exhaustive.

Data structure and sample size

Both `iris` and `penguins` are in tidy format (Wickham, 2014) with each column denoting a single variable and each row containing measurements for a single iris flower or penguin, respectively. The two datasets are comparable in size: dimensions (columns × rows) are 5 × 150 and 8 × 344 for `iris` and `penguins`, respectively, and sample sizes within species are similar (Tables **??** & **??**).

Notably, while sample sizes in `iris` across species are all the same, sample sizes in `penguins` differ across the three species. The inclusion of three factor variables in `penguins` (`species`, `island`, and `sex`), along with `year`, create additional opportunities for grouping, faceting, and analysis compared to the single factor (`Species`) in `iris`.

Unlike `iris`, which contains only complete cases, the `penguins` dataset contains a small number of missing values (n_{missing} = 19, out of 2,752 total values). Missing values and unequal sample sizes are common in real-world data, and create added learning opportunity to the `penguins` dataset.

Continuous quantitative variables

Distributions, relationships between variables, and clustering can be visually explored between species for the four structural size measurements in `penguins` (flipper length, body mass, bill length and depth; Figure 1) and `iris` (sepal width and length, petal width and length; Figure 2).

Both `penguins` and `iris` offer numerous opportunities to explore linear relationships and correlations, within and across species (Figures 1 & 2). A bivariate scatterplot made with the `iris` dataset reveals a clear linear relationship between petal length and petal width. Using `penguins` (Figure 3), we

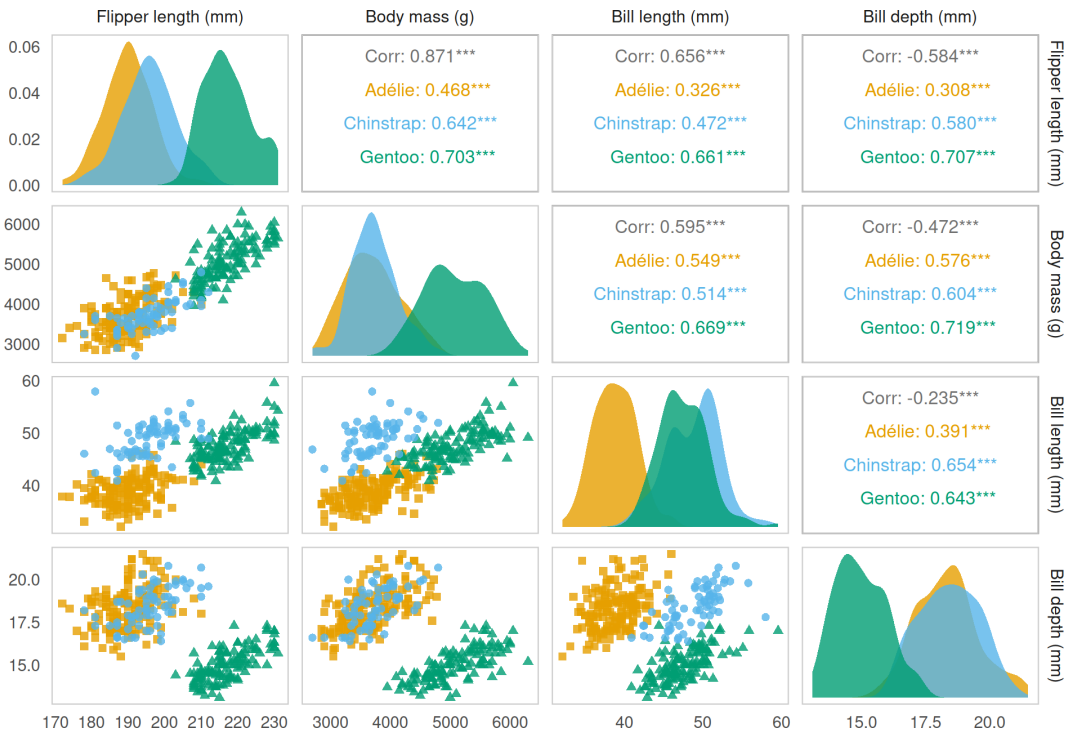


Figure 1: Distributions and correlations for numeric variables in the **penguins** data (flipper length (mm), body mass (g), bill length (mm) and bill depth (mm)) for the three observed species: Gentoo (green, triangles); Chinstrap (blue, circles); and Adélie (orange, squares). Significance indicated for bivariate correlations: *$p < 0.05$; **$p < 0.01$; ***$p < 0.001$.

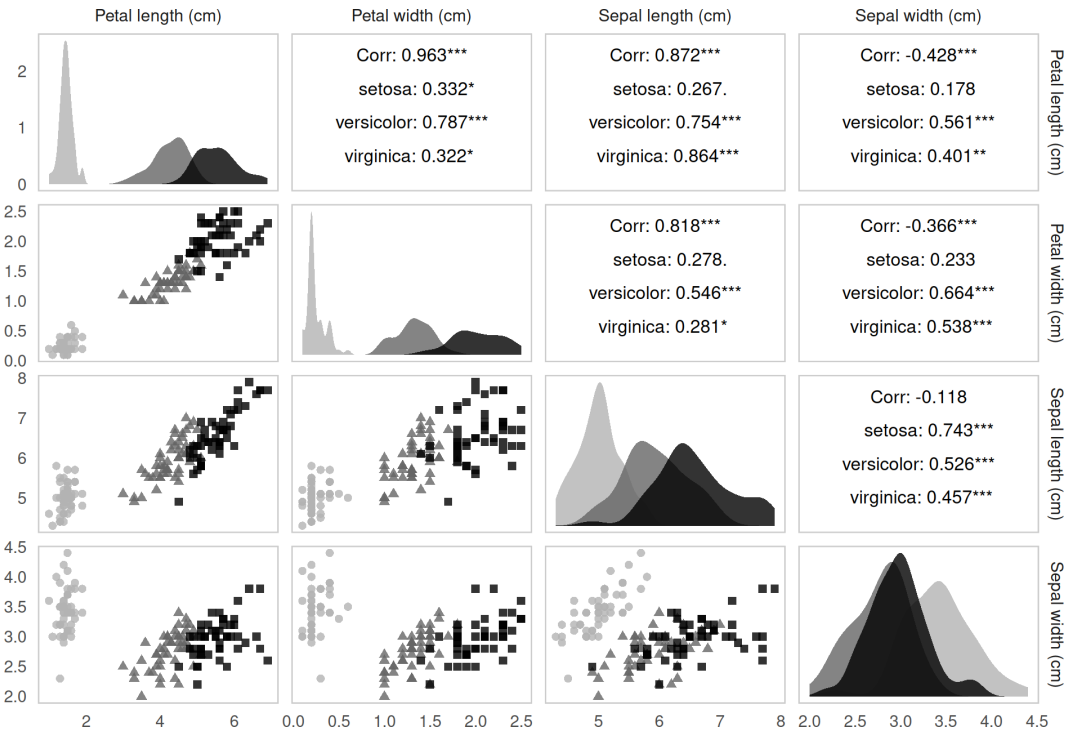


Figure 2: Distributions and correlations for numeric variables in **iris** (petal length (cm), petal width (cm), sepal length (cm) and sepal width (cm)) for the three included iris species: *Iris setosa* (light gray, circles); *Iris versicolor* (dark gray, triangles); and *Iris virginica* (black, squares). Significance indicated for bivariate correlations: *$p < 0.05$; **$p < 0.01$; ***$p < 0.001$.

can create a uniquely similar scatterplot with flipper length and body mass. The overall trend across all three species is approximately linear for both `iris` and `penguins`. Teachers may encourage students to explore how simple linear regression results and predictions differ when the species variable is omitted, compared to, for example, multiple linear regression with species included (Figure 3).

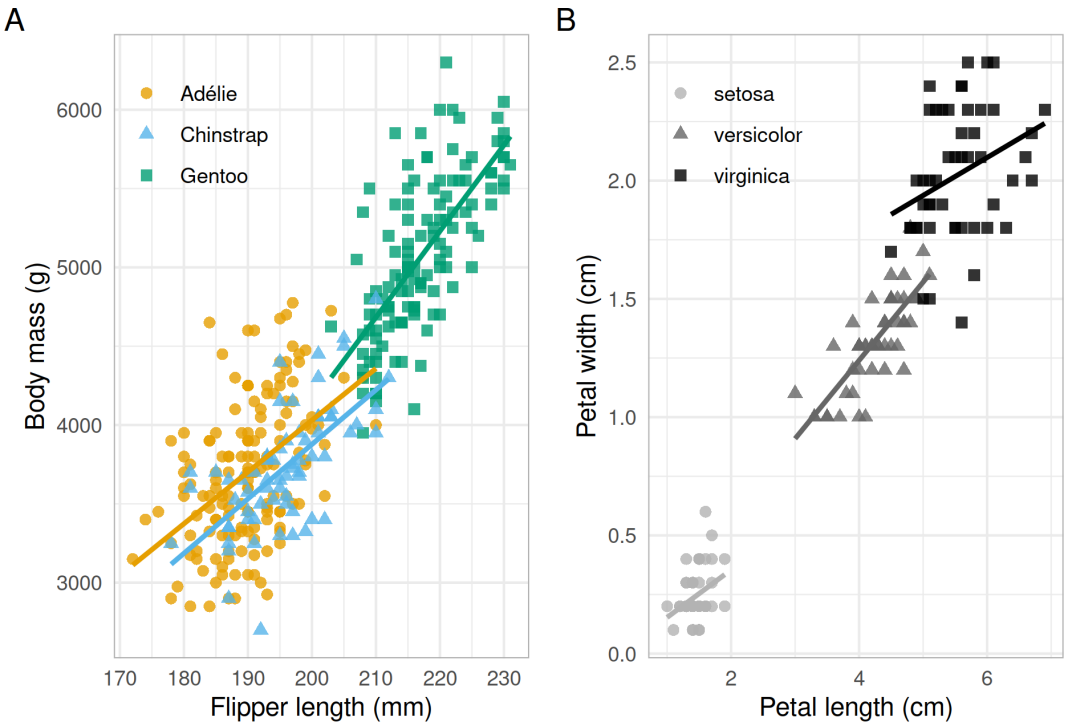


Figure 3: Representative linear relationships for (A): penguin flipper length (mm) and body mass (g) for Adélie (orange circles), Chinstrap (blue triangles), and Gentoo (green squares) penguins; (B): iris petal length (cm) and width (cm) for *Iris setosa* (light gray circles), *Iris versicolor* (dark gray triangles) and *Iris virginica* (black squares). Within-species linear model is visualized for each penguin or iris species.

Notably, distinctions between species are clearer for iris petals - particularly, the much smaller petals for *Iris setosa* - compared to penguins, in which Adélie and Chinstrap penguins are largely overlapping in body size (body mass and flipper length), and are both generally smaller than Gentoo penguins.

Simpson's Paradox is a data phenomenon in which a trend observed between variables is reversed when data are pooled, omitting a meaningful variable. While often taught and discussed in statistics courses, finding a real-world and approachable example of Simpson's Paradox can be a challenge. Here, we show one (of several possible - see Figure 1) Simpson's Paradox example in `penguins`: exploring bill dimensions with and without species included (Figure 4). When penguin species is omitted (Figure 4A), bill length and depth appear negatively correlated overall. The trend is reversed when species is included, revealing an obviously positive correlation between bill length and bill depth within species (Figure 4B).

Principal component analysis

Principal component analysis (PCA) is a dimensional reduction method commonly used to explore patterns in multivariate data. The `iris` dataset frequently appears in PCA tutorials due to multivariate normality and clear interpretation of variable loadings and clustering.

A comparison of PCA with the four variables of structural size measurements in `penguins` and `iris` (both normalized prior to PCA) reveals highly similar results (Figure 5). For both datasets, one species is distinct (Gentoo penguins, and *setosa* irises) while the other two species (Chinstrap/Adélie and *versicolor*/*virginica*) appear somewhat overlapping in the first two principal components (Figure 5 A,B). Screeplots reveal that the variance explained by each principal component (PC) is very similar across the two datasets, particularly for PC1 and PC2: for `penguins`, 88.15% of total variance is captured by the first two PCs, compared to 95.81% for `iris`, with a similarly large percentage of variance captured by PC1 and PC2 in each (Figure 5 C,D).

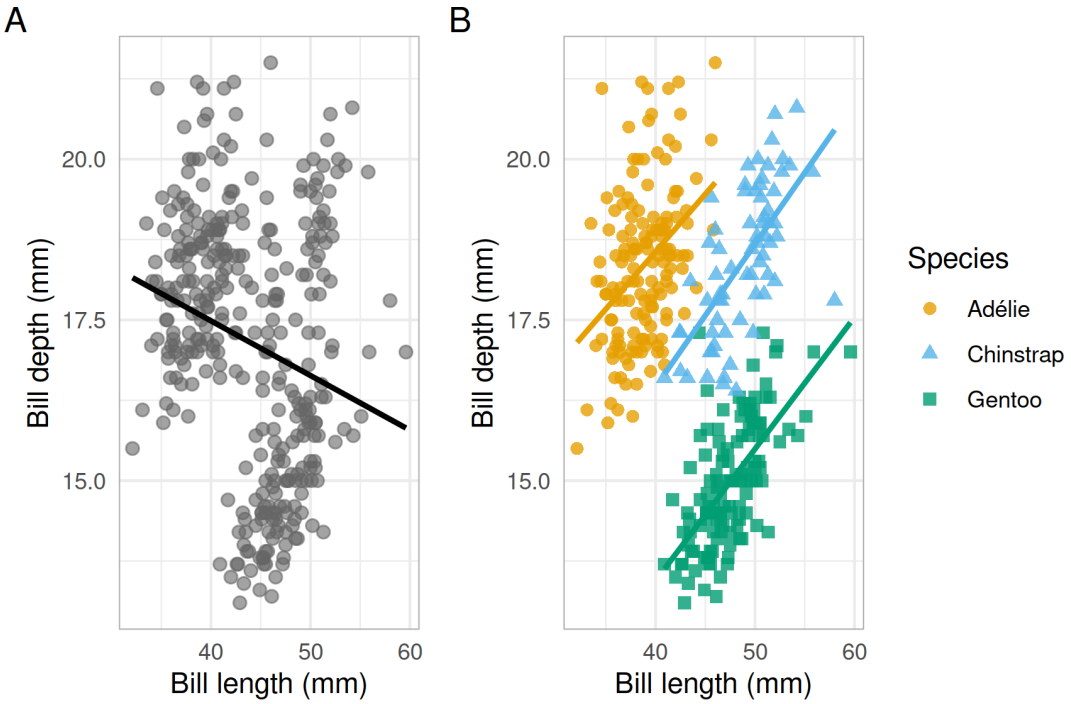


Figure 4: Trends for penguin bill dimensions (bill length and bill depth, millimeters) if the species variable is excluded (A) or included (B), illustrating Simpson's Paradox. Note: linear regression for bill dimensions without including species in (A) is ill-advised; the linear trendline is only included to visualize trend reversal for Simpson's Paradox when compared to (B).

Table 3: K-means cluster assignments by species based on penguin bill length (mm) and depth (mm), and iris petal length (cm) and width (cm).

Penguins cluster assignments				Iris cluster assignments			
Cluster	Adélie	Chinstrap	Gentoo	Cluster	setosa	versicolor	virginica
1	0	9	116	1	0	2	46
2	4	54	6	2	0	48	4
3	147	5	1	3	50	0	0

K-means clustering

Unsupervised clustering by k-means is a common and popular entryway to machine learning and classification, and again, the `iris` dataset is frequently used in introductory examples. The `penguins` data provides similar opportunities for introducing k-means clustering. For simplicity, we compare k-means clustering using only two variables for each dataset: for `iris`, petal width and petal length, and for `penguins`, bill length and bill depth. All variables are scaled prior to k-means. Three clusters ($k = 3$) are specified for each, since there are three species of irises (*Iris setosa*, *Iris versicolor*, and *Iris virginica*) and penguins (Adélie, Chinstrap and Gentoo).

K-means clustering with penguin bill dimensions and iris petal dimensions yields largely distinct clusters, each dominated by one species (Figure 6). For iris petal dimensions, k-means yields a perfectly separated cluster (Cluster 3) containing all 50 *Iris setosa* observations and zero misclassified *Iris virginica* or *Iris versicolor* (Table 3). While clustering is not perfectly distinct for any penguin species, each species is largely contained within a single cluster, with little overlap from the other two species. For example, considering Adélie penguins (orange observations in Figure 6A): 147 (out of 151) Adélie penguins are assigned to Cluster 3, zero are assigned to Cluster 1, and 4 are assigned to the Chinstrap-dominated Cluster 2 (Table 3). Only 5 (of 68) Chinstrap penguins and 1 (of 123) Gentoo penguins are assigned to the Adélie-dominated Cluster 3 (Table 3).

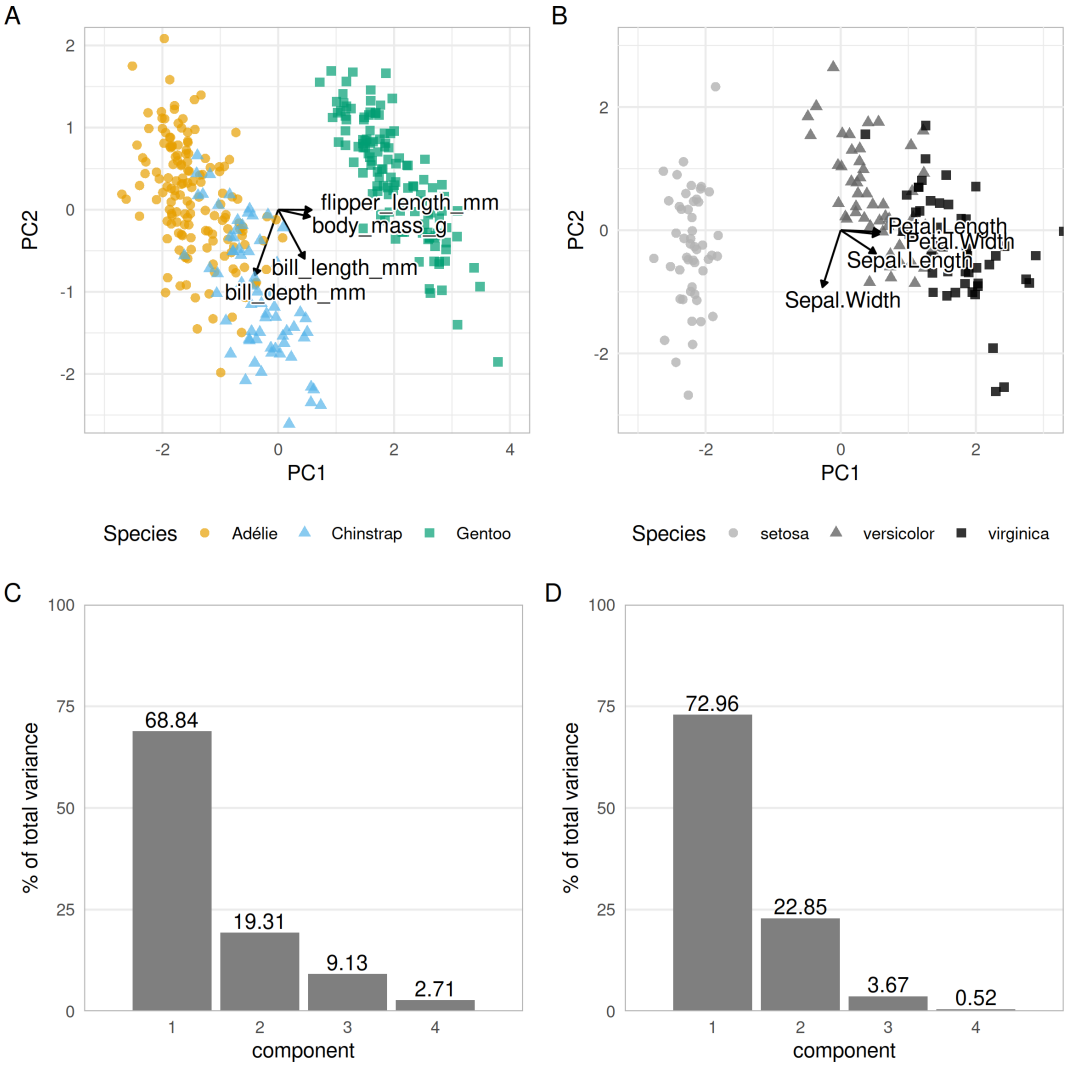


Figure 5: Principal component analysis biplots and screeplots for structural size measurements in **penguins** (A,C) and **iris** (B,D), revealing similarities in multivariate patterns, variable loadings, and variance explained by each component. For **penguins**, variables are flipper length (mm), body mass (g), bill length (mm) and bill depth (mm); groups are visualized by species (Adélie = orange circles, Chinstrap = blue triangles, Gentoo = green squares). For **iris**, variables are petal length (cm), petal width (cm), sepal length (cm) and sepal width (cm); groups are visualized by species (*Iris setosa* = light gray circles, *Iris versicolor* = dark gray triangles, *Iris virginica* = black squares). Values above screeplot columns (C,D) indicate percent of total variance explained by each of the four principal components.

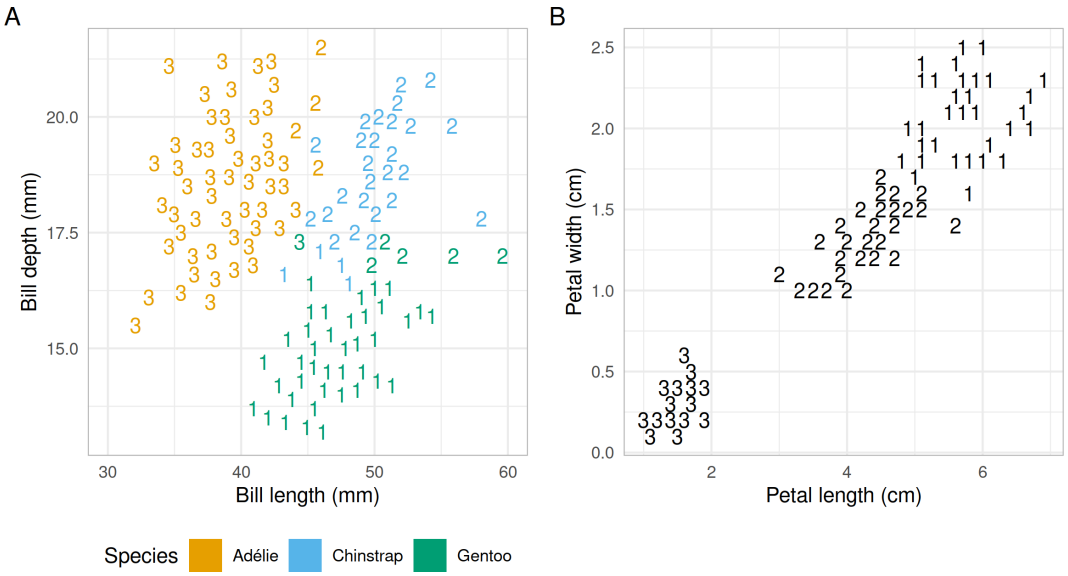


Figure 6: K-means clustering outcomes for penguin bill dimensions (A) and iris petal dimensions (B). Numbers indicate the cluster to which an observation was assigned, revealing a high degree of separation between species for both **penguins** and **iris**.

Conclusion

Here, we have shown that structural size measurements for Palmer Archipelago *Pygoscelis* penguins, available as `penguins` in the **palmerpenguins** R package, offer a near drop-in replacement for `iris` in a number of common use cases for data science and statistics education including exploratory data visualization, linear correlation and regression, PCA, and clustering by k-means. In addition, teaching and learning opportunities in `penguins` are increased due to a greater number of variables, missing values, unequal sample sizes, and Simpson's Paradox examples. Importantly, the `penguins` dataset encompasses real-world information derived from several charismatic marine predator species with regional breeding populations notably responding to environmental change occurring throughout the Western Antarctic Peninsula region of the Southern Ocean (see Bestelmeyer et al. (2011), Gorman et al. (2014), Gorman et al. (2017), Gorman et al. (2021)). Thus, the `penguins` dataset can facilitate discussions more broadly on biodiversity responses to global change - a contemporary and critical topic in ecology, evolution, and the environmental sciences.

Penguins data processing

Data in the `penguins` object have been minimally updated from `penguins_raw` as follows:

- All variable names are converted to lower snake case (e.g. from `Flipper Length (mm)` to `flipper_length_mm`)
- Entries in `species` are truncated to only include the common name (e.g. "Gentoo", instead of "gentoo penguin (*Pygoscelis papua*)")
- Recorded sex for penguin N36A1, originally recorded as ".", is updated to `NA`
- `culmen_length_mm` and `culmen_depth_mm` variable names are updated to `bill_length_mm` and `bill_depth_mm`, respectively
- Class for categorical variables (`species`, `island`, `sex`) is updated to factor
- Variable `year` was pulled from clutch observations

Summary of the `penguins_raw` dataset

Feature	penguins_raw
Year(s) collected	2007 - 2009
Dimensions (col x row)	17 x 344
Documentation	complete metadata
Variable classes	character (9), Date (1), numeric (7)
Missing values?	yes (n = 336; 5.7%)

palmerpenguins for other programming languages

Python: Python users can load the palmerpenguins datasets into their Python environment using the following code to install and access data in the palmerpenguins Python package:

```
pip install palmerpenguins
from palmerpenguins import load_penguins
penguins = load_penguins()
```

Julia: Julia users can access the penguins data in the **PalmerPenguins.jl** package. Example code to import the penguins data through **PalmerPenguins.jl** (more information on **PalmerPenguins.jl** from David Widmann can be found here):

```
julia> using PalmerPenguins
julia> table = PalmerPenguins.load()
```

TensorFlow: TensorFlow users can access the penguins data in TensorFlow Datasets. Information and examples for **penguins** data in TensorFlow can be found here.

Acknowledgements

All analyses were performed in the R language environment using version 4.1.2 (R Core Team, 2021). Complete code for this paper is shared in the Supplemental Material. We acknowledge the following R packages used in analyses, with gratitude to developers and contributors:

- **GGally** (Schloerke et al., 2021): for pairs plots
- **ggiraph** (Gohel and Skintzos, 2022): for interactive **ggplot2** graphics
- **ggplot2** (Wickham et al., 2021): for data visualizations
- **kableExtra** (Zhu, 2021): for finalized tables
- **paletteer** (Hvitfeldt, 2021): for the Okabe Ito color palette, provided by the **colorblindr** package
- **patchwork** (Pedersen, 2020): for compound figures
- **plotly** (Sievert et al., 2021): for interactive graphics
- **recipes** (Kuhn and Wickham, 2021) and **broom** (Robinson et al., 2022): for modeling
- **shadowtext** (Yu, 2022): to add a background color to text labels
- **tidyverse** (Wickham et al., 2019): for data import and cleaning

Bibliography

E. Anderson. The irises of the Gaspé Peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935. [p244]

B. T. Bestelmeyer, A. M. Ellison, W. R. Fraser, K. B. Gorman, S. J. Holbrook, C. M. Laney, M. D. Ohman, D. P. C. Peters, F. C. Pillsbury, A. Rassweiler, R. J. Schmitt, and S. Sharma. Analysis of abrupt transitions in ecological systems. *Ecosphere*, 2(12):art129, Dec. 2011. ISSN 2150-8925. doi: 10.1890/ES11-00216.1. URL http://doi.wiley.com/10.1890/ES11-00216.1. [p251]

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 (2):179–188, Sept. 1936. ISSN 20501420. doi: 10.1111/j.1469-1809.1936.tb02137.x. URL http://doi.wiley.com/10.1111/j.1469-1809.1936.tb02137.x. [p244]

D. Gohel and P. Skintzos. *ggiraph: Make 'ggplot2' Graphics Interactive*, 2022. URL https://CRAN.R-project.org/package=ggiraph. R package version 0.8.2. [p252]

K. B. Gorman, T. D. Williams, and W. R. Fraser. Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus *Pygoscelis*). *PLoS ONE*, 9(3):e90081, Mar. 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0090081. URL https://dx.plos.org/10.1371/journal.pone.0090081. [p245, 251]

K. B. Gorman, S. L. Talbot, S. A. Sonsthagen, G. K. Sage, M. C. Gravely, W. R. Fraser, and T. D. Williams. Population genetic structure and gene flow of Adélie penguins (*Pygoscelis adeliae*) breeding throughout the western Antarctic Peninsula. *Antarctic Science*, 29(6):499–510, Dec. 2017. ISSN 0954-1020, 1365-2079. doi: 10.1017/S0954102017000293. URL https://www.cambridge.org/core/product/identifier/S0954102017000293/type/journal_article. [p251]

K. B. Gorman, K. E. Ruck, T. D. Williams, and W. R. Fraser. Advancing the Sea Ice Hypothesis: Trophic Interactions Among Breeding *Pygoscelis* Penguins With Divergent Population Trends Throughout the Western Antarctic Peninsula. *Frontiers in Marine Science*, 8:526092, Sept. 2021. ISSN 2296-7745. doi: 10.3389/fmars.2021.526092. URL https://www.frontiersin.org/articles/10.3389/fmars.2021.526092/full. [p251]

A. Horst, A. Hill, and K. Gorman. *palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data*, 2020. URL https://CRAN.R-project.org/package=palmerpenguins. R package version 0.1.0. [p244]

E. Hvitfeldt. *paletteer: Comprehensive Collection of Color Palettes*, 2021. URL https://github.com/EmilHvitfeldt/paletteer. R package version 1.3.0. [p252]

M. Kuhn and H. Wickham. *recipes: Preprocessing and Feature Engineering Steps for Modeling*, 2021. URL https://CRAN.R-project.org/package=recipes. R package version 0.1.17. [p252]

Palmer Station Antarctica LTER and K. B. Gorman. Structural size measurements and isotopic signatures of foraging among adult male and female Adélie penguins (*Pygoscelis adeliae*) nesting along the Palmer Archipelago near Palmer Station, 2007-2009, 2020a. URL https://portal.edirepository.org/nis/mapbrowse?packageid=knb-lter-pal.219.5. [p245]

Palmer Station Antarctica LTER and K. B. Gorman. Structural size measurements and isotopic signatures of foraging among adult male and female Gentoo penguin (*Pygoscelis papua*) nesting along the Palmer Archipelago near Palmer Station, 2007-2009, 2020b. URL https://portal.edirepository.org/nis/mapbrowse?packageid=knb-lter-pal.220.5. [p245]

Palmer Station Antarctica LTER and K. B. Gorman. Structural size measurements and isotopic signatures of foraging among adult male and female Chinstrap penguin (*Pygoscelis antarctica*) nesting along the Palmer Archipelago near Palmer Station, 2007-2009, 2020c. URL https://portal.edirepository.org/nis/mapbrowse?packageid=knb-lter-pal.221.6. [p245]

T. L. Pedersen. *patchwork: The Composer of Plots*, 2020. URL https://CRAN.R-project.org/package=patchwork. R package version 1.1.1. [p252]

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. [p244]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/. [p244, 252]

D. Robinson, A. Hayes, and S. Couch. *broom: Convert Statistical Objects into Tidy Tibbles*, 2022. URL https://CRAN.R-project.org/package=broom. R package version 0.7.11. [p252]

B. Schloerke, D. Cook, J. Larmarange, F. Briatte, M. Marbach, E. Thoen, A. Elberg, and J. Crowley. *GGally: Extension to ggplot2*, 2021. URL https://CRAN.R-project.org/package=GGally. R package version 2.1.2. [p252]

C. Sievert, C. Parmer, T. Hocking, S. Chamberlain, K. Ram, M. Corvellec, and P. Despouy. *plotly: Create Interactive Web Graphics via plotly.js*, 2021. URL https://CRAN.R-project.org/package=plotly. R package version 4.10.0. [p252]

H. Wickham. Tidy Data. *Journal of Statistical Software*, 59(10), 2014. ISSN 1548-7660. doi: 10.18637/jss.v059.i10. URL http://www.jstatsoft.org/v59/i10/. [p246]

H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686. [p252]

H. Wickham, W. Chang, L. Henry, T. L. Pedersen, K. Takahashi, C. Wilke, K. Woo, H. Yutani, and D. Dunnington. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2021. URL https://CRAN.R-project.org/package=ggplot2. R package version 3.3.5. [p252]

G. Yu. *shadowtext: Shadow Text Grob and Layer*, 2022. URL https://github.com/GuangchuangYu/shadowtext/. R package version 0.1.1. [p252]

H. Zhu. *kableExtra: Construct Complex Table with kable and Pipe Syntax*, 2021. URL https://CRAN.R-project.org/package=kableExtra. R package version 1.3.4. [p252]

Allison M. Horst
University of California Santa Barbara
Bren School of Environmental Science and Management
Santa Barbara, CA 93106-5131
ahorst@ucsb.edu

Alison Presmanes Hill
Voltron Data

apreshill@gmail.com

Kristen B. Gorman
University of Alaska Fairbanks
College of Fisheries and Ocean Sciences
2150 Koyukuk Drive
245 O'Neill Building
Fairbanks, AK 99775-7220
kbgorman@alaska.edu

Advancing Reproducible Research by Publishing R Markdown Notebooks as Interactive Sandboxes Using the learnr Package

by Chak Hau Michael Tso, Michael Hollaway, Rebecca Killick, Peter Henrys, Don Monteith, John Watkins, and Gordon Blair

Abstract Various R packages and best practices have played a pivotal role to promote the Findability, Accessibility, Interoperability, and Reuse (FAIR) principles of open science. For example, (1) well-documented R scripts and notebooks with rich narratives are deposited at a trusted data centre, (2) R Markdown interactive notebooks can be run on-demand as a web service, and (3) R Shiny web apps provide nice user interfaces to explore research outputs. However, notebooks require users to go through the entire analysis, while Shiny apps do not expose the underlying code and require extra work for UI design. We propose using the **learnr** package to expose certain code chunks in R Markdown so that users can readily experiment with them in guided, editable, isolated, executable, and resettable code sandboxes. Our approach does not replace the existing use of notebooks and Shiny apps, but it adds another level of abstraction between them to promote reproducible science.

Introduction

There has been considerable recognition of the need to promote open and reproducible science in the past decade. The FAIR principles (Wilkinson et al., 2016; Stall et al., 2019) (https://www.go-fair.org/fair-principles/) of reproducible research are now known to most scientists. While significant advances has been made through the adoption of various best practices and policies (e.g. requirements from funders and publishers to archive data and source code, metadata standards), there remains considerable barriers to further advance open science and meet reproducible science needs. One of such issues the availability of various levels of abstraction of the same underlying analysis and code base to collaborate and engage with different stakeholders of diverse needs (Blair et al., 2019; Hollaway et al., 2020). For complex analysis or analysis that utilize a more advanced computing environment, it is essential to provide the capability to allow users to interact with the analysis at a higher level.

Existing approach to reproducible research focuses on either documenting an entire analysis or allows user-friendly interaction. Within the R ecosystem, R scripts and notebooks allow researchers to work together and to view the entire workflow, while R Shiny apps (Chang et al., 2019) allows rapid showcase of methods and research outcomes to users with less experience. R Shiny has been widely adopted to share research output and engage stakeholders since its conception in 2013. A recent review (Kasprzak et al., 2021) shows that bioinformatics is the subject with the most Shiny apps published in journals while earth and environmental science ranks second. Shiny apps are especially helpful to create reproducible analysis (e.g. examples in Hollaway et al., 2020) and explore different scenarios (e.g. Whateley et al., 2015; Mose et al., 2018). Finally, the interactivity of Shiny apps makes it an excellent tool for teaching (e.g. Williams and Williams, 2017; Field, 2020). However, not all users fit nicely into this dichotomy. Some users may only want to adopt a small fraction of an analysis for their work, while others may simply want to modify a few parts of the analysis in order to test alternative hypothesis. Current use of notebooks do not seem to support such diverse needs as notebook output *elements* (e.g. figures and tables) are not easily reproducible. This issue essentially applies to all coding languages.

One potential way to address the problem described above is to allow users to experiment with the code in protected computing environment. This is not limited to creating instances for users to re-run the entire code. Rather, this can also be done by exposing specific parts of a notebook as editable and executable code boxes, as seen in many interactive tutorial web pages for various coding languages. Recently, while discussing next steps for fostering reproducible research in artificial intelligence, Carter et al. (2019) lists creating a protected computing environment ('data enclave' or 'sandbox') for reviewers to log in and explore as one of the solutions. In software engineering, a sandbox is a testing environment that isolates untested code changes and outright experimentation from the production environment or repository, in the context of software development including Web development and revision control. Making a sandbox environment available for users to test and explore various changes to the code that leads to research outputs is a great step to further open

science. Current practice of open science largely requires users to assemble the notebooks, scripts and data files provided in their own computing environment, which requires significant amount of time and effort. A sandbox environment can greatly reduce such barriers and if such sandboxes are available as a web service, users can explore and interact with the code that generates the research outputs at the convenience of their own web browser on demand.

In this paper, we describe a rapid approach to create and publish 'interactive sandboxes' R Shiny apps from R Markdown documents using the **learnr** package, with the aim to bridge the gap between typical R Markdown notebook and typical Shiny apps in terms of levels of abstraction. While code and markdown documents gives full details of the code, standard R Shiny apps has too much limitations on users to interact with the code and users often cannot see the underlying code. Our approach allows users to interact with selected parts of the code in an isolated manner, by specifying certain code chunks in a R Markdown document as executable code boxes.

The *learnr* R package

learnr (Schloerke et al., 2020) is an R package developed by RStudio to rapidly create interactive tutorials. It follows the general *R Markdown* (the file has .Rmd extensions, `https://rmarkdown.rstudio.com/index.html`) architecture and essentially creates a pre-rendered Shiny document similar to the way Shiny user interface (UI) components can be added to any R Markdown documents. Pre-rendered Shiny documents (`https://rmarkdown.rstudio.com/authoring_shiny_prerendered.HTML`) is a key enabling technology for the **learnr** package since it allows users to specify the execution context in each code chunk of a R Markdown document that is used to render a R Shiny web app. Its use circumvents the need of a full document render for each end user browser session so that this type of R Shiny apps can load quickly. To create a **learnr** tutorial in RStudio after **learnr** is installed, the user chooses a **learnr** R Markdown template from a list after clicking the "create new R Markdown document" button. This template is not different from other .Rmd files, except it requires additional chunk arguments to control the sandbox appearances. The two main features of the **learnr** package are the "exercise" and "quiz" options. The former allows users to directly type in code, execute it, and see its results to test their knowledge while the latter allows other question types such as multiple choice. Both of these options include auto-graders, hints, and instructor feedback options. Additional overall options include setting time limits and an option to forbid users to skip sections. Like any Shiny apps, **learnr** apps can be easily embedded to other web pages, as seen in Higgins (2021).

Although the **learnr** package has existed for a few years now, it is relatively not well known to scientists as a potential use of R Shiny Apps and it has mostly been used for simple tutorial apps designed for R beginners. We propose a novel application of the **learnr** package to advance reproducible research, which we outline in the next section.

Approach: Using *learnr* for reproducible research 'sandboxes'

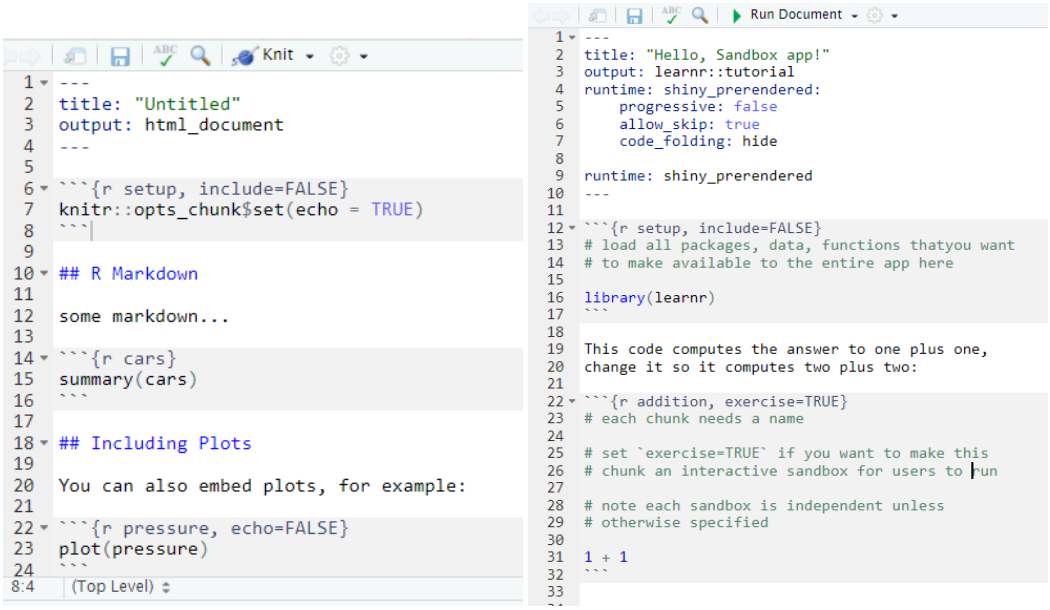
learnr allows users to create executable code boxes. Our approach is to publish R notebooks and serve parts of the notebooks as interactive sandboxes to allow users to re-create certain elements of a published notebook containing research outputs. We do not use the auto-graders or any quiz-like functionality of **learnr** while keeping the sandboxes. Notebook authors can go through their notebook and select the code chunks that they would allow users to experiment, while the others are rendered as static code snippets.

Recognizing **learnr** documents are themselves R Shiny web apps, our approach essentially allows the publication of notebooks in the form of web apps. However, unlike a typical R Shiny web app, users do not need to prepare a separate UI (i.e. user interface) layout. Advanced users can modify the site appearance by supplying custom design in .css files.

Here, we first show the skeleton of a R Markdown (.Rmd) file for a **learnr** document (Figure 1). Notice that it is very similar to a typical .Rmd file where there is a mixture of narratives written in markdown and R code chunks, in addition to a YAML header. However, there are a couple of important exceptions, namely the use of the "exercise" chunk option (i.e. editable and executable code boxes) and different output type in the YAML header.

Next, we outline the steps an author needs to take to publish notebooks (i.e. R Markdown documents) as interactive sandboxes:

1. All research output is included in the form of a well-documented R Markdown document.
2. Open a new **learnr** R Markdown template. Copy the content of the original notebook.
3. For the code chunks that you would like to become sandboxes, add `exercise=TRUE`. Make sure it has a unique chunk name. It may look something like this:



(a) A typical .Rmd file **(b)** A sandbox app .Rmd file

Figure 1: A comparison of minimal examples of a typical .Rmd document and a .Rmd document for an interactive sandbox app.

```
```{r fig2, warning=FALSE, exercise=TRUE, exercise.lines=30,fig.fullwidth=TRUE}
```

4. Before any interactive code chunks, call the first code chunk 'setup'. This will pre-load everything that will be used later.
5. Check whether you would like to link any of the interactive code snippets (by default each of them are independent, and only depends on the output of the 'setup' chunk) You may want to modify your code chunks accordingly.
6. Done! Knit the notebook to view outputs as an interactive web page. Publish it just like a Shiny app.

The entire process took us a few hours of effort and can be incorporated to the proof-reading of an R Markdown document. However, we note that as in any preparation of research output or web development several iterations are often needed and the time required increases accordingly as the complexity of the analysis increases.

In our implementation in DataLabs (https://datalab.datalabs.ceh.ac.uk/), the environment and folder to create the research is made available to the Shiny app in a read-only fashion. Therefore, the authors do not have to worry about versions of packages of data or a different software setup. Using DataLabs straightforward visual tools to publish R Shiny apps, we can publish an R Markdown notebook with interactive code snippets to reproduce certain parts of research readily in a few clicks.

Deployment

In general, **learnr** tutorial apps can be published the same way as R Shiny web apps in Shiny servers, such as the ones provided by cloud service providers or https://shinyapps.io. The **learnr** package vignettes provide additional help on deployment.

We also describe our deployment of these apps in DataLabs, a UK NERC virtual research environment that is being developed. DataLabs is a collaborative virtual research environment (Hollaway et al., 2020) (https://datalab.datalabs.ceh.ac.uk/) for environmental scientist to work together where data, software, and methods are all centrally located in projects. DataLabs provide a space for scientists from different domains (data science, statisticians, environmental science and computer science) to work together and draw on each other's expertise. It includes an easy-to-use user interface where users can publish R Shiny apps with a few clicks, and this applies to these notebooks with interactive code chunks as well. Importantly, when provisioning a instance of R Shiny, this is deployed in a Docker container with read-only access to the project data store being used for analysis. This allows an unprecedented level of transparency as parts of the analysis are readily exposed for users to experiment from the exact environments, datasets (can be large and includes many files), and versions
```

of software that created the analysis. The use of Docker deployed onto a Kubernetes infrastructure allows strict limits to be placed on what visitors can do through the use of resource constraints and tools such as **RAppArmor** (Ooms, 2013). While access to project files is read-only, some author discretion is still advised to ensure that visitors should not be able to view or list any private code or data. We also note that future releases of **learnr** will contain external exercise evaluators, so that the code sandboxes can be executed by an independent engine (such as Google Cloud) and give the benefit of not having to rely on **RAppArmor**.

Example: GB rainfall paper

To demonstrate our concept, we have turned an R Markdown notebook for one of our recent papers (Tso et al., 2022) into a **learnr** site (`https://cptecn-sandboxdemo.datalabs.ceh.ac.uk/`) using the procedures described in the previous sections. The paper investigates the effect of weather and rainfall types on rainfall chemistry in the UK. As can be seen in Figure 2, the code chunks to generate certain parts of the paper is exposed. But unlike a static notebook site, the code chunk is not only available for copy and paste but allows users to modify and run on-demand. This makes it very straightforward for user to experiment with various changes of the original analysis, thereby promoting transparency and trust.

Since **learnr** apps are R Markdown documents, Shiny UI elements can be easily added. We repeat one of the examples by replacing the interactive code box by a simple selector, with minimal modification of the code itself. This approach to publish Shiny apps requires significantly less work than typical R Shiny web apps since no UI design is needed and researchers can rapidly turn an R Markdown document to an R Shiny web app. For some cases, the use of certain datasets may require a license, as in this example. A pop-up box is shown when the site is loaded and visitors are required to check the boxes to acknowledge the use of the appropriate data licenses (an alternative is to require users to register and load a token file) before they can proceed.

Evaluation

The main strength of our approach is that it fills nicely the gap of existing approaches in terms of levels of abstraction. While code and markdown documents gives full details of the code, standard R Shiny apps has too much limitations on users to interact with the code (Figure 3) and users often cannot see the underlying code. Recently, it has become popular to publish 'live' Jupyter notebooks on Binder and Google Colab. While this is a great contribution to open science, users are still required to run and go through the entire notebook step-by-step and it can be easy to break it if users change something in between. Our approach allows users to interact with portions of the code in a guided and isolated manner, without the need to understand all the other parts of a notebook or the fear to break it (Table 1). We emphasize that R scripts/notebooks and R Shiny apps work well for their intended uses, but our approach adds an additional level of accessibility to users.

The openness and ease-to-access our approach provides can benefit many different stakeholders (Table 2). Researchers can more rapidly reproduce *parts* of the analysis of their choice without studying the entire notebook or installing software or downloading all the data. They can quickly test alternative hypothesis and stimulate scientific discussions. For funders, encouraging the use of this approach means less time is needed for future projects to pick up results from previous work. And since this is based on **learnr** which is originally designed as a tutorial tool, this approach will no doubt speed up the process to train other users to use similar methods. Overall, it promotes open science and make a better value of public funds.

An obvious limitation of our approach is that it does not work well for ideal conditions where other R file formats are designed for. For instance, R scripts and R notebooks are much better suited for more complex analysis for users to adopt to their own problems. Meanwhile, R Shiny web apps provides a much richer user experience and is most suited when the exposed code is generally not useful to stakeholders. Nevertheless, as discussed above, our approach is designed for users to reproduce *elements* of an analysis. The user should evaluate these options carefully, paying special attention to the needs of intended users.

Serving notebooks as a web service will inevitably face provenance issues. It is surely beneficial if the author's institution can host these interactive notebooks for a few years after its publication (and that of its related publications). In the future, publishers and data centres may consider providing services to provide longer term provenance of serving these interactive notebooks online. As for any web apps, funding for the computation servers can be a potential issue. This work uses DataLabs computation time which is part of the UK research funding that develops it. However, a more rigorous funding model may be needed in the future to ensure provenance of these notebooks.

GB rainfall chemistry

Welcome

Figure 1: ECN map

Figure 1: ECN map (with selectors)

Figure 2: Lamb cyclones/westerlies counts (1871-2020)

Figure 3: Lamb vs chemistry boxplots

Start Over

Figure 3: Lamb vs chemistry boxplots

This code plots concentration over time for selected chemical species across ECN sites grouped by number of Lamb westerlies and cyclones per week. Chemical samples are taken weekly.

Code | Start Over | Run Code

```
  if(skip_legend == 1){
    out = out + theme(legend.position = "none")
  }else{
    out = out +  theme(legend.position = "bottom",legend.justification = 'center',
                       legend.key.size=unit(36,units = "points") ,
                       legend.text = element_text(size = 18))
  }
  return(out)
}

grp_name = 'lamb_grp2'

p1 = plot_chembox(date_chem, "CONDY","lamb_grp2") + xlab('Year')+ylab('Electrical Conductivity (\u03BCS/cm)')
p4 = plot_chembox(date_chem, "NO3N","lamb_grp2") + xlab('Year')+ylab( expression("NO"[3]^-{}*" (\u03BCeq/L)"))
p5 = plot_chembox(date_chem, "NH4N","lamb_grp2") + xlab('Year')+ylab( expression("NH"[4]^+{}*" (\u03BCeq/L)"))+
  theme(legend.position = "bottom",legend.justification = 'center',
        legend.key.size=unit(36,units = "points") ,
        legend.text = element_text(size = 18)) +
  guides(fill=guide_legend("Number of Westerly days per week: " ))
p2 = plot_chembox(date_chem, "CHLORIDE","lamb_grp2") + xlab('Year')+ylab( expression("Cl"[]^-{}*" (\u03BCeq/L)"))+
  theme(legend.position = "bottom",legend.justification = 'center',
        legend.key.size=unit(36,units = "points") ,
        legend.text = element_text(size = 18)) +
  guides(fill=guide_legend("Number of Westerly days per week: " ))
p3 = plot_chembox(date_chem, "SODIUM","lamb_grp2") + xlab('Year')+ylab( expression("Na"[]^+{}*" (\u03BCeq/L)"))
p6 = plot_chembox(date_chem, "non-marine sulphate","lamb_grp2") + xlab('Year')+ylab( expression("xSO"[4]^2-{}*" (\u03BCeq/L)")
pALL = plot_grid(p1,p2,p3,labels = c("A","B","C"), align = 'h', axis = "tb",nrow = 1)
pALL
pALL = plot_grid(p4,p5,p6,labels = c("D","E","F"), align = 'h', axis = "tb",nrow = 1)
```

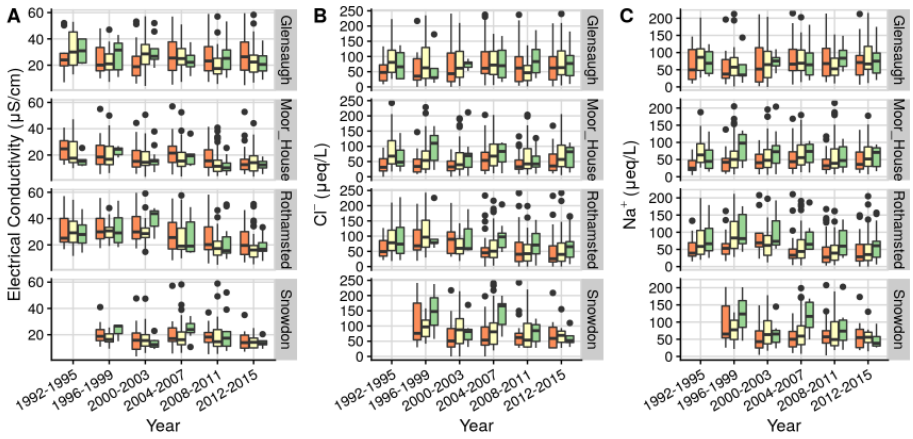


Figure 2: A screenshot of the GB rainfall interactive notebook site. The main feature is the code box. When the site loads, the code that generates published version of the figure is in the box and published version of the figure is below it. Users can make edits and re-run the code in the code box and the figure will update accordingly. Users can use the "Start Over" button to see the published version of the code at any point without refreshing the entire site.

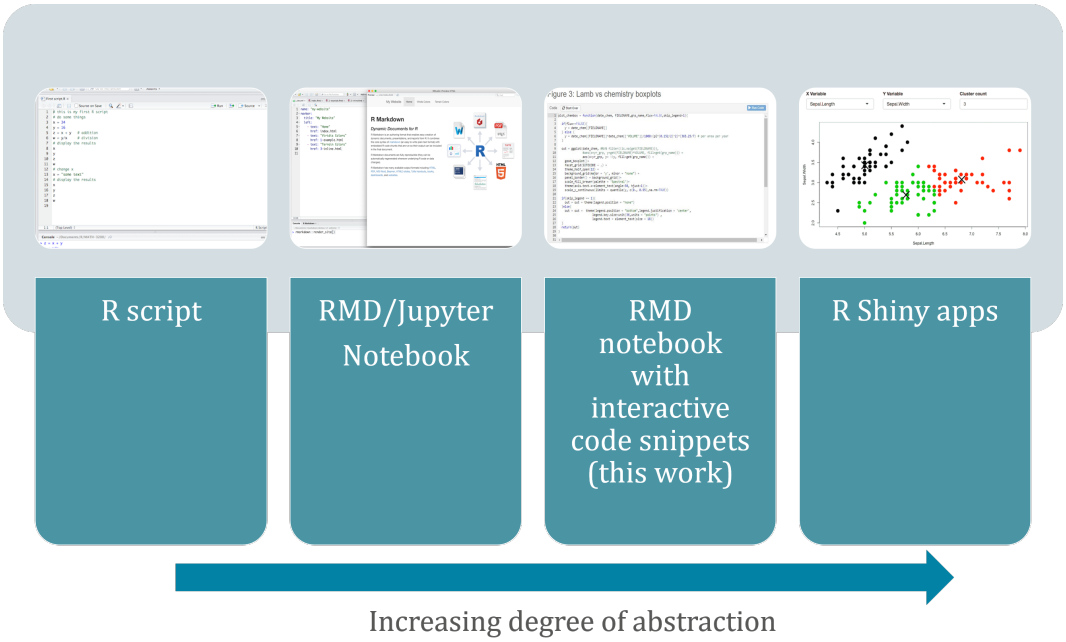


Figure 3: The various levels of abstraction of various types of R documents. Our approach fills nicely the gap between R Markdown or Jupyter notebooks and Shiny apps.

Our approach focuses on improving reproducibility by exposing parts of R script for users to run them live on an R Shiny web app, leveraging the option to render R Markdown documents as R Shiny web apps and the **learnr** package. It focuses on the R scripts and R Markdown documents. Users, however, may want to improve reproducibility from the opposite direction, namely to allow outputs from an R Shiny web app to be reproducible outside of the Shiny context. For such a requirement, we recommend the use of the **shinymeta** (Cheng and Sievert, 2021) package, which allows users to capture the underlying code of selected output elements and allows users to download it as well as the underlying data to re-create the output in their own R instance. The **shinymeta** approach can be more involved and requires more effort than **learnr** so we think it is more suitable for users that are focusing their effort on the R Shiny app (particularly the UI). In summary, these two approaches complements each other and we recommend users to consider them to improve reproducibility of their work.

Summary and outlook

We have proposed and demonstrated a rapid approach to publish R Markdown notebooks as interactive sandboxes to allow users to experiment with changes with various elements of a research output. It provides an additional level of abstraction for users to interact with research outputs and the codes that generates down. Since it can be linked to the environment and data that generated the published output and has independent document object identifiers (DOI), it is a suitable candidate to preserve research workflow while exposing parts of it to allow rapid experimentation by users. Our work is a demonstration on how we may publish a notebook from virtual research environments such as DataLabs, with data, packages, and workflow pre-loaded in a coding environment, accompanied by rich narratives. While this paper outlines the approach using R, the same approach can benefit other coding languages such as Python. In fact, this can already be achieved as **learnr** can run Python chunks (as well as other execution engines **knitr** supports such as SAS and mySQL) as long as the users generate and host the document using R. This paper contributes to the vision towards publishing interactive notebooks as standalone research outputs and the advancement of open science practices.

Data availability and acknowledgements

The GB rainfall example notebook is accessible via this URL (`https://cptecn-sandboxdemo.datalabs.ceh.ac.uk/`) and the R Markdown file is deposited in the NERC Environmental Information Data Centre (EIDC) (Tso, 2022). The DataLab code stack is available at `https://github.com/NERC-CEH/datalab`. We thank the DataLabs developers team (especially Iain Walmsley, UKCEH) for the assistance to deploy interactive R Markdown documents on DataLabs. This work is supported by NERC Grant NE/T006102/1, Methodologically Enhanced Virtual Labs for Early Warning of Significant

Table 1: Advantages of the proposed approach to various stakeholders

Advantages
Authors • Very little extra work required in additional to writing R markdown document. • No experience to generate web interfaces required. • Much greater impact in research output.
Other researchers (those wanting to try or compare the method) • A much more enriched experience to try methods and data and to test alternative hypothesis and scenarios. • No need to download data and scripts/notebooks and install packages to try a method. • More efficient to learn the new method.
Other researchers (those curious about the results) • Try running different scenarios quickly than the published ones without the hassle of full knowledge of the code, downloading the code and data, and setting up the software environment. • Quickly reset to the published version of code snippet. • No need to worry about breaking the code.
Data Centres • A new avenue to demonstrate impact to funders if end users try methods or datasets hosted by them in sandboxes.
Funders • Better value of investment if even small parts of a research is readily reproducible. • Time saving to fund related work that builds on research documented this way.
Wider research community and general public • Promotes trust and confidence in research through transparency.

or Catastrophic Change in Ecosystems: Changepoints for a Changing Planet, funded under the Constructing a Digital Environment Strategic Priority Fund. Additional support is provided by the UK Status, Change and Projections of the Environment (UK-SCAPE) programme started in 2018 and is funded by the Natural Environment Research Council (NERC) as National Capability (award number NE/R016429/1). The initial development work of DataLabs was supported by a NERC Capital bid as part of the Environmental Data Services (EDS).

Bibliography

G. S. Blair, P. Henrys, A. Leeson, J. Watkins, E. Eastoe, S. Jarvis, and P. J. Young. Data Science of the Natural Environment: A Research Roadmap. *Frontiers in Environmental Science*, 7, 2019. ISSN 2296-665X. doi: 10.3389/fenvs.2019.00121. URL https://www.frontiersin.org/article/10.3389/fenvs.2019.00121/full. [p255]

R. E. Carter, Z. I. Attia, F. Lopez-Jimenez, and P. A. Friedman. Pragmatic considerations for fostering reproducible research in artificial intelligence. *npj Digital Medicine*, 2(1), 2019. doi: 10.1038/s41746-019-0120-2. URL https://doi.org/10.1038/s41746-019-0120-2. [p255]

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2019. URL https://CRAN.R-project.org/package=shiny. R package version 1.4.0. [p255]

J. Cheng and C. Sievert. *shinymeta: Export Domain Logic from Shiny using Meta-Programming*, 2021. URL https://CRAN.R-project.org/package=shinymeta. R package version 0.2.0.1. [p260]

A. Field. *adventr: Interactive R Tutorials to Accompany Field (2016), "An Adventure in Statistics"*, 2020. URL https://cran.r-project.org/web/packages/adventr/index.html. [p255]

P. Higgins. *Reproducible Medical Research with R*, 2021. URL https://bookdown.org/pdr_higgins/rmrwr/. [p256]

Table 2: Advantages of the proposed approach over existing approaches

Potential Issues	How our approach can help?
R script	
• Limited narrative. Needs to run all the scripts.	• Much richer narrative and interactive experience.
Static notebooks	
• Needs to download the code, data and package to try it out.	• Can instantly try out the code in a controlled manner, using the published data/packages/software environment.
Web apps (e.g. Shiny)	
• While web apps helpful to some stakeholders, it can be too high-level to some.	• Users can interact with the code within the code snippet sandboxes themselves.
• Lots of extra work to create web interface.	• The published version of the code is shown to users.
• Does not expose the code to generate results.	• Users can run the code snippets live.
Binder/Google Colab	
• Users change the entire notebook.	• A much more enriched and guided experience.
• Users need to run all cells about the section they are interested in.	• Users can choose to only run the sandboxes they are interested in.

M. J. Hollaway, G. Dean, G. S. Blair, M. Brown, P. A. Henrys, and J. Watkins. Tackling the challenges of 21st-century open science and beyond: A data science lab approach. *Patterns*, 1(7):100103, 2020. doi: 10.1016/j.patter.2020.100103. URL https://doi.org/10.1016/j.patter.2020.100103. [p255, 257]

P. Kasprzak, L. Mitchell, O. Kravchuk, and A. Timmins. Six years of shiny in research - collaborative development of web tools in R. *R Journal*, (3):155–162, 2021. doi: 10.32614/RJ-2021-004. URL https://journal.r-project.org/archive/2021/RJ-2021-009/RJ-2021-009.pdf. [p255]

V. N. Mose, D. Western, and P. Tyrrell. Application of open source tools for biodiversity conservation and natural resource management in east africa. *Ecological Informatics*, 47:35–44, Sept. 2018. doi: 10.1016/j.ecoinf.2017.09.006. URL https://doi.org/10.1016/j.ecoinf.2017.09.006. [p255]

J. Ooms. The RAppArmor package: Enforcing security policies in R using dynamic sandboxing on linux. *Journal of Statistical Software*, 55(7):1–34, 2013. URL http://www.jstatsoft.org/v55/i07/. [p258]

B. Schloerke, J. Allaire, and B. Borges. *learnr: Interactive Tutorials for R*, 2020. URL https://CRAN.R-project.org/package=learnr. R package version 0.10.1. [p256]

S. Stall, L. Yarmey, J. Cutcher-Gershenfeld, B. Hanson, K. Lehnert, B. Nosek, M. Parsons, E. Robinson, and L. Wyborn. Make scientific data FAIR, 2019. ISSN 14764687. URL https://doi.org/10.1038/d41586-019-01720-7. [p255]

C.-H. M. Tso. R shiny sandbox app demonstrator for advancing reproducible research. NERC Environmental Information Data Centre. (Model). https://doi.org/10.5285/df57b002-2a42-4a7d-854f-870dd867618c, 2022. URL https://doi.org/10.5285/df57b002-2a42-4a7d-854f-870dd867618c. [p260]

C.-H. M. Tso, D. Monteith, T. Scott, H. Watson, B. Dodd, M. G. Pereira, P. Henrys, M. Hollaway, S. Rennie, A. Lowther, J. Watkins, R. Killick, and G. Blair. The evolving role of weather types on rainfall chemistry under large reductions in pollutant emissions. *Environmental Pollution*, 299: 118905, 2022. ISSN 02697491. doi: 10.1016/j.envpol.2022.118905. URL https://doi.org/10.1016/j.envpol.2022.118905. [p258]

S. Whateley, J. D. Walker, and C. Brown. A web-based screening model for climate risk to water supply systems in the northeastern united states. *Environmental Modelling & Software*, 73:64–75, 2015. doi: 10.1016/j.envsoft.2015.08.001. URL https://doi.org/10.1016/j.envsoft.2015.08.001. [p255]

M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J. W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S. A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. Van Der Lei, E. Van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons. Comment: The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 2016. ISSN 20524463. doi: 10.1038/sdata.2016.18. URL https://doi.org/10.1038/sdata.2016.18. [p255]

I. J. Williams and K. K. Williams. Using an R shiny to enhance the learning experience of confidence intervals. *Teaching Statistics*, 40(1):24–28, 2017. doi: 10.1111/test.12145. URL https://doi.org/10.1111/test.12145. [p255]

Chak Hau Michael Tso
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
ORCiD: 0000-0002-2415-0826
mtso@ceh.ac.uk

Michael Hollaway
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
ORCiD: 0000-0003-0386-2696
mhollaway@ceh.ac.uk

Rebecca Killick
Department of Statistics, Lancaster University
Flyde College
Lancaster LA1 4YF, United Kingdom
ORCiD: 0000-0003-0583-3960
r.killick@lancaster.ac.uk

Peter Henrys
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
ORCiD: 0000-0003-4758-1482
pehn@ceh.ac.uk

Don Monteith
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
donm@ceh.ac.uk

John Watkins
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
ORCiD: 0000-0002-3518-8918
jww@ceh.ac.uk

Gordon Blair
UK Centre for Ecology and Hydrology
Lancaster Environment Centre
Lancaster LA1 4YQ, United Kingdom
ORCiD: 0000-0001-6212-1906
g.blair@lancaster.ac.uk

Power and Sample Size for Longitudinal Models in R – The longpower Package and Shiny App

by Samuel Iddi and Michael C. Donohue

Abstract Longitudinal studies are ubiquitous in medical and clinical research. Sample size computations are critical to ensure that these studies are sufficiently powered to provide reliable and valid inferences. There are several methodologies for calculating sample sizes for longitudinal studies that depend on many considerations including the study design features, outcome type and distribution, and proposed analytical methods. We briefly review the literature and describe sample size formulas for continuous longitudinal data. We then apply the methods using example studies comparing treatment versus control groups in randomized trials assessing treatment effect on clinical outcomes. We also introduce a Shiny app that we developed to assist researchers with obtaining required sample sizes for longitudinal studies by allowing users to enter required pilot estimates. For Alzheimer's studies, the app can estimate required pilot parameters using data from the Alzheimer's Disease Neuroimaging Initiative (ADNI). Illustrative examples are used to demonstrate how the package and app can be used to generate sample size and power curves. The package and app are designed to help researchers easily assess the operating characteristics of study designs for Alzheimer's clinical trials and other research studies with longitudinal continuous outcomes. Data used in preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu).

1 Introduction

Longitudinal designs are generally preferred over cross-sectional research designs as they provide richer data and greater statistical power. As such, many biomedical and medical studies employ longitudinal designs to study changes over time in outcomes at the individual, group, or population level. Early in the design of a longitudinal experimental or natural history study, it is imperative to ensure that the study is adequately powered for its aims. Inadequate sample sizes leads to invalid or inconclusive inference and squandered resources (Lu et al., 2009; Yan and Su, 2006). On the other hand, oversampling squanders resources and exposes participants to unnecessary risks associated with the intervention (Lu et al., 2009). Thus, optimal sample size and power analysis have become important prerequisites for any quantitative research design. Not only are these required during the design phase of research, but it has also become mandatory for ethical, scientific, or economic justification in submissions to institutional review boards and research funding agencies.

Determining the right sample size for a study is not a straightforward task. Despite the plethora of sample size formulas for repeated measures (Overall and Doyle, 1994; Lui, 1992; Rochon, 1991; Guo et al., 2013), cluster repeated measures (Liu et al., 2002), multivariate repeated measures (Vonesh and Schork, 1986; Guo and Johnson, 1996), longitudinal research designs (Lefante, 1990), the tasks of gathering pilot estimates of the necessary parameters and getting the right software to carry out the computation can be challenging. Researchers commonly rely on formulas for very basic cross-sectional studies and adjust for attrition and other longitudinal design effects. Although such approaches can yield appropriate approximations, the ideal approach is to use a formula derived directly from the longitudinal model that researchers plan to eventually use on the study data.

Guo et al. (2013) describe practical methods for the selection of appropriate sample size for repeated measures addressing issues of missing data, and the inclusion of more than one covariate to control for differences in response at baseline. Sample size formulas are refined depending on the specific situation and design features. For example, Hedeker et al. (1999) considered a sample size for longitudinal designs comparing two groups that accounted for participant attrition or drop-out. Basagaña et al. (2011) derived sample size formulas for continuous longitudinal data with time-varying exposure variables typical of observational studies. Ignoring time-varying exposure was demonstrated to lead to substantial overestimation of the minimum required sample size which can be economically disadvantageous. In non-traditional longitudinal designs such as designs for mediation analysis of the longitudinal study, further refinements to sample size formulas are needed to ensure that sufficient sample sizes are obtained (Pan et al., 2018). However, these formulas usually require additional parameters such as exposure mean, variance, and intraclass correlations (Basagaña et al., 2011), mediation effect, number of repeated measures (Pan et al., 2018), covariance structures (Rochon, 1991), non-linear trends (Yan and Su, 2006), missing, attrition or dropout rates (Roy et al., 2007; Lu

et al., 2008), among others. Advanced sample size methods simultaneously handle several practical issues associated with the research design and complications that may arise during data collection. However, such methods are only available in commercial software (NCSS Statistical Software, 2021; nQuery, 2021).

Several R packages can be found on CRAN to compute sample size based on mixed-effect models and other specific designs depending on the area of applications. For example, Martin et al. (2011) proposed a simulation-based power calculation and an R package **pamm** for random regression models, a specific form of mixed-effect model that detects significant variation in individual or group slopes. In their approach, a power analysis was performed to detect a specified level of individual and environmental interactions within evolution and ecology applications. This is achieved by simulating power to detect a given covariance structure. Other simulation-based packages for power analysis are the **SIMR** by Green and MacLeod (2016) for linear and generalized linear mixed models and **clusterPower** by Kleinman (2021) for cluster-randomized and cross-over designs. Schoenfeld (2019) developed a power and sample size package called **LPower** (Diggle et al., 1994) to perform power analysis for longitudinal design accounting for attrition and different random effect specification. The approach requires the specification of a design matrix, and the variance-covariance matrix of the repeated measures (Yi and Panzarella, 2002). In pharmacokinetic study designs, Kloprogge and Tarning (2015) developed the **PharmPow** power calculation package for mixed study designs including crossover and parallel designs. Quite recently, other R packages for performing power analysis exist for different designs; for example the **powerMediation** (Qiu, 2021) for mediation effect, mean change for longitudinal study with 2-time points, the slope for simple Poisson regression, etc.; **powerEQTL** (Dong et al., 2021) for unbalanced one-way ANOVA in a Bulk Tissue and Single-Cell eQTL Analysis; **WebPower** (Zhang et al., 2021) for basic and advanced power correlation, proportion, *t*-test, one-way ANOVA, two-way ANOVA, linear regression, logistic regression, Poisson regression, mediation analysis, longitudinal data analysis, structural equation modeling, and multilevel modeling; among others. These packages were tailored towards very specific areas of application although the methods can be adapted and utilized for other disciplines. In terms of software applications, attempts have been made in recent times to implement power and sample size calculations in Shiny applications to facilitate easy usage. For example, Hemming et al. (2020) developed an R Shiny app to conduct power analysis for several cluster designs including parallel, cross-over, and stepped-wedge designs. Schoemann et al. (2017) Shiny application conducts power analysis for mediation model, and Hu and Qu (2021) performs sample size and power calculations for a random coefficient regression model (RCRM) and a two-stage mixed-effects model.

As the analysis model and associated sample size formula become more sophisticated, estimating the parameters required by the formula becomes more challenging. A major hurdle to overcome is the availability of pilot data to inform these parameters. To assist Alzheimer's researchers with this challenge, we have developed a power and sample size Shiny app for Alzheimer's clinical trials. The app implements formulas for the linear mixed-effects model and mixed model for repeated measures (MMRM; Lu et al., 2008) allowing the user to input their pilot estimates, or allow the app to generate pilot estimates using data from the Alzheimer's Disease Neuroimaging Initiative (ADNI; Weiner et al., 2015).

2 Review of methods

Continuous outcomes in clinical trial data collected longitudinally over time are commonly analyzed using linear mixed models (LMM; Laird and Ware, 1982) or MMRM (Mallinckrodt et al., 2001, 2003). Before such trials, it is necessary to estimate the required sample size for a given treatment effect with desired power and Type I error. Various sample size approaches for longitudinal data have been proposed. We review a few of the most commonly used methods applied in Alzheimer's disease trials with continuous outcomes.

Sample size computation based on the linear mixed-effects model

Several sample size approaches have been developed by different authors. Diggle et al. (2002) proposed sample size formulas for two approaches to continuous longitudinal outcomes, one that assumes a constant mean over time and compares the average response over time between groups, and another that assumes a linear change over time and compares the mean rate of change or slope difference between groups. In either case, the null hypothesis is that there is no difference between groups. Suppose that Y_{ij} is the response for participant i taken at time j. To compare the average response between two groups (e.g. group A receiving active treatment and group B receiving control) across time, consider the model

$$Y_{ij} = \beta_0 + \beta_1 T_i + \epsilon_{ij}$$

where T_i is the treatment indicator which is coded 1 for participants in group A and 0 for participants in group B, and ϵ_{ij} the error term for subject, i at time j, which is normally distributed with common variance, σ^2. The sample size required per group to detect an average response between the two groups, δ, is given by

$$m = \frac{2(Z_{1-\alpha/2} + Z_P)^2\sigma^2(1 + (J-1)\rho)}{J\delta^2}$$

where α is the type I error rate, P is the level of power, and $\text{corr}(Y_{ij}, Y_{ik}) = \rho$ for all $j \neq k$.

To test difference in the slopes or average rate of change between the two groups, consider the parameterization:

$$Y_{ij} = \beta_{0A} + \beta_{1A}t_{ij} + \epsilon_{ij}, i = 1, 2, \ldots, m; j = 1, 2, \ldots, J.$$

A similar parameterization can be specified for group B with parameters β_{0B}, and β_{1B} representing the intercept and slope. The parameters β_{1A} and β_{1B} are the rate of change of the outcome for groups A and B, respectively. Assume that $\text{var}(\epsilon_{ij}) = \sigma^2$ and $\text{corr}(Y_{ij}, Y_{ik}) = \rho$ for all $j \neq k$. If the outcome of each participant is measured at the same time, t_j, then the sample size per group needed to detect a difference in the rate of change for each group, $\Delta = \beta_A - \beta_B$, is given by

$$m = \frac{2(Z_{1-\alpha/2} + Z_P)^2\sigma^2(1-\rho)}{Js_t^2\Delta^2}$$

where α is the type I error rate, P is the power, and $s_t^2 = \frac{\sum(t_j - \bar{t})}{J}$ is the within-participant variance of t_j.

Another sample size computation approach for correlated data is derived by Liu and Liang (1997) to detect differences in the average response between two groups. This approach derived sample size following the generalized estimating equation (Liang and Zeger, 1986) approach. Thus, different outcomes types can be handled. A special case is for a continuous response measured repeatedly over time and modeled using a linear model. Consider the model

$$Y_{ij} = \beta_0 + \beta_1 T_i + \epsilon_{ij}, i = 1, 2, \ldots, N, j = 1, 2, \ldots, J$$

where T_i is the treatment indicator, ϵ_{ij} is the error term assumed to follow a multivariate normal distribution with a mean vector of zeros, $\mathbf{0}$, and covariance matrix, $\sigma^2\boldsymbol{R}$ and $\boldsymbol{R}$ is a $J \times J$ working correlation matrix. Different covariance structures such as independent, exchangeable, auto-regressive, and unstructured can be assumed. To estimate the total sample size required, N for a given significance level (α) and pre-specified nominal power, P, the following formula is used:

$$N = \frac{(Z_{1-\alpha/2} + Z_P)^2\,\sigma^2}{\pi_A\pi_B(\beta_1^2\mathbf{1}\boldsymbol{R}^{-1}\mathbf{1})}$$

where π_A and π_B represent the proportion of observations in the control and treated groups, respectively. By assuming an exchangeable working correlation matrix, this formula reduces to

$$N = \frac{(Z_{1-\alpha/2} + Z_P)^2\,\sigma^2[1 + (J-1)\rho]}{\pi_A\pi_B\beta_1^2 J}.$$

It can be observed that for equal proportion of participants in each group i.e. $\pi_A = \pi_B = \frac{1}{2}$, the formula for the sample size per group is equivalent to the sample size formula for difference in average response between groups by Diggle et al. (2002) above. An appealing aspect of Liu and Liang (1997) approach is that it can allow for unequal allocation to the two groups, and different outcome types are considered.

Another approach is based on the LMM analysis (Fitzmaurice et al., 2004) comparing mean rate of change between groups (Ard and Edland, 2011; Zhao and Edland, 2020). Consider the LMM

$$Y_{ij} = X_{ij}'\boldsymbol{\beta} + b_{0i} + b_{1i}t_{ij} + \epsilon_{ij}$$

where X_{ij} denotes a vector of indicator variables (treatment, time, and interaction terms), $\boldsymbol{b}_i = (b_{0i}, b_{1i})' \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{bmatrix} \sigma_{b_0}^2 & \sigma_{b_0,b_1} \\ \sigma_{b_0,b_1} & \sigma_{b_1}^2 \end{bmatrix}\right)$ are random participant-specific effects, and $\epsilon_{ij} \sim N(0, \sigma_e^2)$ are the residual errors. The random terms, $\boldsymbol{b}_i$ and ϵ_{ij} are assumed to be independent of each other. Following this model specification, Ard and Edland (2011) proposed a total sample size formula given by

$$N = \frac{4\,(Z_{1-\alpha/2} + Z_P)^2}{\Delta^2}\left[\sigma_{b_1}^2 + \frac{\sigma_e^2}{\sum(t_j - \bar{t})^2}\right]$$

where t_j indexes the times at which measures are made assuming all participants are observed at the same time points, $\bar{t}$ is the mean of the times, $\sigma^2_{b_1}$ is the variance of the participant-specific random slope, and Δ is the difference in mean rate of change in treatment versus control group to be detected. For a random-intercept mixed-effect model, the total sample size formula reduces to

$$N = \frac{4\,(Z_{1-\alpha/2} + Z_P)^2}{\Delta^2}\left[\frac{\sigma^2_e}{\sum(t_j - \bar{t})^2}\right].$$

This sample size formula does not account for the correlation between the random intercept and slope, a limitation that can be mitigated by a recent approach by Hu et al. (2021).

Sample size computation based on the mixed model for repeated measures (MMRM)

An alternative approach is to treat time as a categorical variable. This approach, common in trials of treatments of Alzheimer's and other therapeutic areas, is often referred to as the MMRM (Mallinckrodt et al., 2001, 2003; Lane, 2008). The approach to fitting the MMRM is similar to that for other linear mixed-effect models for longitudinal or repeated measures except the unstructured modelling of time – treated as a categorical variable, and the specification of a within-participant correlation structure to account for association among the repeated measurements. The MMRM provides an estimate of the mean response for each time point category, for each group, and the resulting mean trajectory over time is unconstrained. The primary test statistic is usually the estimated group difference at the final time point. The null hypothesis is again that there is no difference between groups.

Suppose that Y_{ij} is the jth measurement taken of participant $i; i = 1, 2, \ldots, N$ at time, t_{ij} and T_i represents treatment or group indicator. Assuming we have two groups, $a; = (A, B)$, let n_{a_j} be the number of participants at time $j = 1, 2, \ldots, J$ for group a. Thus, at time point 1, the total number of participants is $n_{A1} + n_{B1}$. The response variable is assumed to follow a multivariate normal distribution with mean, variance-covariance and correlation matrix given respectively as

$$\begin{aligned} E(Y_{ij}|T_i = a) &= \mu_{aj} \\ cov(Y_{ij}, Y_{ik}|T_i = a) &= \sigma_{ajk} = \boldsymbol{\Sigma}_a \\ corr(Y_{ij}, Y_{ik}|T_i = a) &= \frac{\sigma_{jk}}{\sigma_j \sigma_k} = \boldsymbol{R} \end{aligned}$$

where $\sigma_j = \sigma_{jj}^{\frac{1}{2}}$. It is common to assume that the variance-covariance matrix is the same for each group in which case we can have $\boldsymbol{\Sigma}_A = \boldsymbol{\Sigma}_B = \boldsymbol{\Sigma}$. Different correlation structures, $\boldsymbol{R}$, can be assumed (eg., compound symmetry, autoregressive and unstructured). The sample size formula for MMRM (Lu et al., 2008) can account for the attrition rate at each time point. Define the attrition rate $\zeta_{aj} = 1 - r_{aj}$, where $r_{aj} = \frac{n_{aj}}{n_{a1}}$ is the retention rate at time point, j. The sample size for group A needed to achieve a power, P, at a significance level, α is given by

$$n_{A1} = (\psi_A + \lambda\psi_B)(Z_{1-\alpha/2} + Z_P)^2 \frac{\sigma_J^2}{\delta_J}$$

where $\lambda = \frac{n_{A1}}{n_{B1}}$ is called the allocation ratio, δ_J is the effect size (difference in mean response between the two groups) at the last time point, J, and ψ_a is the (J, J)th component of the variance inflation factor, I_a^{*-1} defined with respect to an estimation of the μ_{aj} for treatment group, a. Specifically, $\psi_a = \left[I_a^{*-1}\right]_{JJ}$ where

$$I_a^* = \sum_{j=1}^{J}(r_{a,j} - r_{a,j+1}) \begin{bmatrix} \boldsymbol{R}_j^{-1} & 0_{j\times(J-j)} \\ 0_{(J-j)\times j} & 0_{(J-j)\times(J-j)} \end{bmatrix}$$

The total sample size at the first time point is therefore given by

$$n_{A1} + n_{B1} = (1 + \lambda^{-1})(\psi_A + \lambda\psi_B)(Z_{1-\alpha/2} + Z_P)^2 \frac{\sigma_J^2}{\delta_J}.$$

The longpower package

The aforementioned sample size approaches have been implemented in an R package, **longpower** (Donohue and Edland, 2020), that can be found on CRAN via the URL `https://cran.r-project.org/web/packages/longpower/index.html`. The package also contains functions which translate pilot mixed effect model parameters (e.g. random intercept and/or slope) into marginal model parameters

so that the formulas of Diggle et al. (2002) or Liu and Liang (1997) or Lu et al. (2008) formula can be applied to produce sample size calculations for two sample longitudinal designs assuming known variance.

A web-based app

The interactive Shiny (Chang et al., 2021) application available from the URL `https://atrihub.shinyapps.io/power/` is an interface to the **longpower** package developed to easily generate sample size and conduct power analysis for a longitudinal study design with two-group comparisons for a continuous outcome. The app similarly implements the sample size formula of Liu and Liang (1997) and Diggle et al. (1994, 2002) using functions in the **longpower** package. A novel feature of the app is that it can generate required pilot estimates by sourcing data from the Alzheimer's Disease Neuroimaging Initiative (ADNI).

3 Illustrations

Illustration of the longpower package

ADNI is a population-based longitudinal cohort study that follows study participants to collect data on their clinical, cognitive, imaging including MRI and PET images, genetic, and biochemical biomarkers. The study was designed to discover, optimize, standardize, and validate clinical trial measures and biomarkers that are used in AD clinical research. This multi-site longitudinal study runs at about 63 sites in the US and Canada and began in 2004. All the data generated from the ADNI study are entered into a data repository hosted at the Laboratory of Neuroimaging (LONI) at the University of Southern California, the LONI Image & Data Archive (IDA). The data can be freely accessed upon request. Apart from the many uses of the data for advancing knowledge for AD trials (Weiner et al., 2015), this big data resource can be used to improve study design. Specifically, in this paper, the data is used to generate pilot estimates for the computation of sample size and power.

We consider an Alzheimer's disease example using ADAS-Cog (Rosen et al., 1984) pilot estimates from the ADNI database. Suppose that we want to compute the sample size required to detect an effect of 1.5 at the 5% level of significance and 80% power. The LMM fit to ADNI data has an estimated variance of random slope for group A (placebo or control group) of 74, and a residual variance of 10. Assuming study visits at (0.00, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50) years, the sample size using the 'edland' approach can be obtained by using the `edland.linear.power()` functions in the **longpower** R package as follows.

```
> t = seq(0,1.5,0.25)
> edland.linear.power(delta=1.5, t=t, sig2.s = 24, sig2.e = 10, sig.level=0.05,
power = 0.80)

Zhao and Edland, in process

N = 414.6202
n = 207.3101, 207.3101
delta = 1.5
t = 0.00, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50
p = 0, 0, 0, 0, 0, 0, 1
p_2 = 0, 0, 0, 0, 0, 0, 1
sig2.int = 0
sig.b0b1 = 0
sig2.s = 24
sig2.e = 10
sig2.int_2 = 0
sig.b0b1_2 = 0
sig2.s_2 = 24
sig2.e_2 = 10
sig.level = 0.05
power = 0.8
alternative = two.sided

NOTE: N is *total* sample size and n is sample size in *each* group
```

An alternative approach is to use `lmmpower()` and specify the argument `method="edland"` in the following way.

```
> lmmpower(delta=1.5, t=t, sig2.s = 24, sig2.e = 10, sig.level=0.05,
power = 0.80,method="edland")

Zhao and Edland, in process

N = 414.6202
n = 207.3101, 207.3101
delta = 1.5
t = 0.00, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50
p = 0, 0, 0, 0, 0, 0, 1
p_2 = 0, 0, 0, 0, 0, 0, 1
sig2.int = 0
sig.b0b1 = 0
sig2.s = 24
sig2.e = 10
sig2.int_2 = 0
sig.b0b1_2 = 0
sig2.s_2 = 24
sig2.e_2 = 10
sig.level = 0.05
power = 0.8
alternative = two.sided

NOTE: N is *total* sample size and n is sample size in *each* group
```

The Diggle and Liu & Liang approaches can be applied with the `diggle.linear.power()` and `lui.liang.linear.power()` functions, respectively. The `lmmpower()` functions can be used for either approach with the appropriate specification of the `` `method' `` argument.

The second illustrative example is the hypothetical clinical trial discussed in Diggle et al. (2002). Suppose that we are interested in testing the effect of a new treatment in reducing blood pressure through a clinical trial. The investigator is interested in randomizing participants between a control and active treatment group to have equal size. Three visits are envisaged with assessments planned at years 0, 2, and 5. Thus, $n = 3$ and $s_x^2 = 4.22$. Assuming a type I error rate of 5%, power of 80% and smallest meaningful difference of 0.5 mmHg/year, we want to determine the number of participants needed for both treated and control groups for varying correlations (0.2, 0.5 and 0.8) and response variance (100, 200, 300).

```
> n <- 3
> t <-  c(0,2,5)
> rho <-  c(0.2, 0.5, 0.8)
> sigma2 <-  c(100, 200, 300)
> tab = outer(rho, sigma2,
+             Vectorize(function(rho, sigma2){
+               ceiling(diggle.linear.power(
+                 delta=0.5,
+                 t=t,
+                 sigma2=sigma2,
+                 R=rho,
+                 alternative="one.sided",
+                 power = 0.80)$n[1])}))
> colnames(tab) = paste("sigma2 =", sigma2)
> rownames(tab) = paste("rho =", rho)
> tab
sigma2 = 100 sigma2 = 200 sigma2 = 300
rho = 0.2          313          625          938
rho = 0.5          196          391          586
rho = 0.8           79          157          235
```

The above code reproduces the table on page 29 of Diggle et al. (2002). We also reproduce the table on page 30 of Diggle et al. (2002) for detecting a difference in average response between two groups through the method by Liu and Liang (1997) as follows:

```
> u = list(u1 = rep(1,n), u2 = rep(0,n)) #a list of covariate vectors or
   matrices associated with the parameter of interest
> v = list(v1 = rep(1,n), v2 = rep(1,n)) #a respective list of covariate
vectors or matrices associated with the nuisance parameter
> rho = c(0.2, 0.5, 0.8) #correlations
> delta = c(20, 30, 40, 50)/100 #effect size
> tab = outer(rho, delta,Vectorize(function(rho, delta){
+   ceiling(liu.liang.linear.power(
+     delta=delta, u=u, v=v,
+     sigma2=1,
+     R=rho, alternative="one.sided",
+     power=0.80)$n[1])}))
> colnames(tab) = paste("delta =", delta)
> rownames(tab) = paste("rho =", rho)
> tab
delta = 0.2 delta = 0.3 delta = 0.4 delta = 0.5
rho = 0.2         145          65          37          24
rho = 0.5         207          92          52          33
rho = 0.8         268         120          67          43
```

The sample size formula for the MMRM approach is also implemented in the **longpower** package's `power.mmrm()` function. To illustrate how this approach is implemented, consider a hypothetical example with a correlation matrix having 0.25 as off-diagonal entries (exchangeable), a retention vector (1, 0.90,0.80,0.70) and standard deviation of 1, for group A. Assuming these values to be the same for group B, then the sample size required to detect an effect size of 0.5 at 5% level of significance and 80% power is computed as follows:

```
> Ra <- matrix(0.25, nrow = 4, ncol = 4)
> diag(Ra) <- 1 #exchangeable correlation matrix for group A
> ra <- c(1, 0.90, 0.80, 0.70)#retention in group A
> sigmaa <- 1 #standard deviation for group A
> power.mmrm(Ra = Ra, ra = ra, sigmaa = sigmaa, delta = 0.5, power = 0.80)

Power for Mixed Model of Repeated Measures (Lu, Luo, & Chen, 2008)

n1 = 86.99175
n2 = 86.99175
retention1 = 1.0, 0.9, 0.8, 0.7
retention2 = 1.0, 0.9, 0.8, 0.7
delta = 0.5
sig.level = 0.05
power = 0.8
alternative = two.sided
```

Suppose the allocation ratio is 2, then the function argument `lambda=2` can be added.

Illustration of the Shiny app

We demonstrate how the app is used to perform power analysis by inputting user-specified values and a specific sample size approach. We make use of the values shown in Table 1. For the MMRM method, we specify the options assuming an exchangeable correlation structure.

For the ADNI-based pilot estimate generator, we select the full range of values for some selected variables (See Figure 3).

App menus and contents

The app has three main menus on the sidebar. The first menu has two sub-menus which let the user perform power and sample size calculations based on longitudinal designs when necessary pilot estimates are known. The first submenu, the default, accepts user-specified inputs and generates a graphical output and summary of the inputs. In the 'input' box, as shown in Figure 1a, users are aided with the selection of their inputs through widgets such as select input, numerical input, slider input, and radio button to facilitate the user selection. The select input for 'Analysis Type' allows the user to select whether the user desires to determine power or sample size. Input parameters that are not

Table 1: Parameter values considered in Shiny app demonstrations. Each row represents a different parameter and the values considered for the linear mixed model (LMM, left column) and the mixed model for repeated measures (MMRM, right column) demonstrations. The "-" indicates the parameter is "not applicable" for the given model.

Parameter	LMM	MMRM
Start time	0	1
End time	1.5	4
Timestep	0.5	1
Type I error rate	0.05	0.05
Effect size	1.5	-
Estimate of variance of random intercept	55	-
Estimate of variance of random slope	22	-
Estimate of covariance of random intercept and slope	29	-
Estimate of the error variance	10	-
Standard deviation of observation in group A	-	1
Standard deviation of observation in group B	-	1
Exchangeable correlation	-	0.25
Allocation ratio	1	2

applicable for a given selection or are not applicable for the selected sample size method are grayed out to enhance the user experience. For example, the allocation ratio input widget is grayed out when the 'diggle' and 'liuliang' procedures are selected. The 'output' box, shown in Figure 1b, displays a graph of power versus sample size, a note on the method for the sample size computation, and a summary of the selected inputs by the user. The second submenu enables the user to conduct power analysis based on the MMRM methodology. Widgets such as select inputs, slider inputs, numerical inputs, and radio buttons display the current values of the model parameter in the 'input' box (See Figure 2a). As the MMRM requires the specification of an association structure and a vector of the retention rates, the app display vectors, and matrix inputs widgets. The size of these widgets depends on users' choice of the number of time points for the study. However, these widgets are not reactive, and therefore the user must use the "Update/Enter" action button when changes are made to the number of time points to update the vector and matrix input widgets. The corresponding graphical and summary outputs are reactively displayed within the 'output' box, immediately below the 'input' box (See Figure 2b).

The second menu item also has two sub-menus for the LMM and the MMRM methods, respectively. In this menu, the app enables the user to generate pilot estimates from the Alzheimer's Disease Neuroimaging Initiative (ADNI) data that is fed into the application to perform the power calculations.

In the box for 'baseline selection', the user can select variables that define their study population. By default, all variables are selected, utilizing all the data in the ADNI. A variable can be deselected from the left side of the box with a click. After the user selection of the variable, the user can submit by clicking on the 'submit selected criteria' action button to activate the corresponding widgets in the next box. The 'Inclusion/exclusion criteria' box is made of a slider and select input widgets which allow the user to select a range of values for the selected population characteristics (See Figure 3). Baseline summaries are produced by the app according to the selections of the user. Summaries by gender, education, ethnicity, and race are provided for the selected population. For continuous variables, the number of observations, number of missing observations, mean, median, lower and upper quantiles are displayed, while for categorical variables, the total and level-specific number of observations, and percentages are displayed. These bivariate summaries give the user a clear presentation of the data per the selected inclusion and exclusion criteria.

Next, the user can choose a primary outcome from among some that are commonly used for Alzheimer's disease studies. Pilot estimates are obtained from fitting a LMM for the selected outcome for the power analysis. The estimates of the model can be adjusted with some user-selected covariate options. Data on the selected outcome from baseline to the selected number of years of the study are presented by individual-level and smooth mean profile plots. The app allows the selection of options for selecting the sample size method, type of test, type I error rate, percentage change, and allocation ratio as inputs for the power analysis (see Figure 4). Finally, the graphical output for the power analysis and summary of the inputs used are displayed in the 'output' box. The summary of the LMM is also shown (see Figure 5).

The interface of the second sub-menu is very similar to the first except that the sample size methodology is based on the MMRM. Additionally, the model fitting options include specifying an

association structure, allocation ratio, and percentage retention for the two groups.

The final menu on the sidebar is the 'About' menu, which provides brief information of the dashboard, data description, and acknowledgment of the ADNI resources, packages used for developing the dashboard, and contact information of the developers of the dashboard.

4 Discussion

In this manuscript, we have presented the **longpower** R package, and a Shiny app dashboard that facilitates sample size and power analysis for a longitudinal study design with two-group comparisons of a continuous outcome. The app implements the sample size formulas of Liu and Liang (1997), Diggle et al. (1994, 2002), Lu et al. (2008), and Ard and Edland (2011) using functions in the **longpower** package. The package also handles models in which time is treated either as continuous (e.g. with random intercepts and slopes) or categorical (MMRM).

The **longpower** package was created to allow R users easy access to sample size formulas for longitudinal data that were already available in the literature. Many of the earlier papers on the topic provided no software, and so considerable effort was required by each reader to program implementations of the formulas. Collecting these formulas into an R package makes the methods more accessible and easy to compare. The package includes unit tests to ensure the software can adequately reproduce published results, and alternative approaches for the same study design are validated against each other.

A novel feature of the app is the ability to source pilot data for Alzheimer's disease trials to generate required parameter estimates. We focus on Alzheimer's data as our primary area of interest, but future work could bring in data from other disease areas. Other future directions include accommodating other outcome types, and keeping up with the evolving landscape of model parameterizations.

Acknowledgments

We are grateful to the ADNI study volunteers and their families. This work was supported by Biomarkers Across Neurodegenerative Disease (BAND-14-338179) grant from the Alzheimer's Association, Michael J. Fox Foundation, and Weston Brain Institute; and National Institute on Aging grant R01-AG049750. Data collection and sharing for this project was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012). ADNI is funded by the National Institute on Aging, the National Institute of Biomedical Imaging and Bioengineering, and through generous contributions from the following: AbbVie, Alzheimer's Association; Alzheimer's Drug Discovery Foundation; Araclon Biotech; BioClinica, Inc.; Biogen; Bristol-Myers Squibb Company; CereSpir, Inc.; Eisai Inc.; Elan Pharmaceuticals, Inc.; Eli Lilly and Company; EuroImmun; F. Hoffmann-La Roche Ltd and its affiliated company Genentech, Inc.; Fujirebio; GE Healthcare; IXICO Ltd.; Janssen Alzheimer Immunotherapy Research & Development, LLC.; Johnson & Johnson Pharmaceutical Research & Development LLC.; Lumosity; Lundbeck; Merck & Co., Inc.; Meso Scale Diagnostics, LLC.; NeuroRx Research; Neurotrack Technologies; Novartis Pharmaceuticals Corporation; Pfizer Inc.; Piramal Imaging; Servier; Takeda Pharmaceutical Company; and Transition Therapeutics. The Canadian Institutes of Health Research is providing funds to support ADNI clinical sites in Canada. Private sector contributions are facilitated by the Foundation for the National Institutes of Health (www.fnih.org). The grantee organization is the Northern California Institute for Research and Education, and the study is coordinated by the Alzheimer's Therapeutic Research Institute at the University of Southern California. ADNI data are disseminated by the Laboratory for Neuro Imaging at the University of Southern California.

Data used in preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf

Conflict of Interest: None declared.

Bibliography

M. C. Ard and S. D. Edland. Power calculations for clinical trials in alzheimer's disease. *J Alzheimers Dis.*, 26(Suppl 3):369–377, 2011. doi: doi:10.3233/JAD-2011-0062. [p266, 272]

X. Basagaña, X. Liao, and D. Spiegelman. Power and sample size calculations for longitudinal studies estimating a main effect of a time-varying exposure. *Statistical methods in medical research*, 20:5, 2011. doi: https://doi.org/10.1177/0962280210371563. [p264]

W. Chang, J. Cheng, J. J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, J. Dipert, and B. Borges. *shiny: Web Application Framework for R*, 2021. URL https://CRAN.R-project.org/package=shiny. R package version 1.6.0. [p268]

P. Diggle, K.-Y. Liang, , and S. L. Zeger. *Analysis of Longitudinal Data*. Oxford University Press, New York, 1994. [p265, 268, 272]

P. Diggle, P. Heagerty, K. Liang, and S. Zeger. *Analysis of longitudinal data. Second Edition*. Oxford Statistical Science Series, 2002. [p265, 266, 268, 269, 272]

X. Dong, X. Li, T.-W. Chang, S. T. Weiss, and W. Qiu. *'powerEQTL': Power and Sample Size Calculation for Bulk Tissue and Single-Cell eQTL Analysis*, 2021. URL https://cran.r-project.org/web/packages/powerEQTL/powerEQTL.pdf. R package version 0.3.4. [p265]

M. C. Donohue and S. D. Edland. *'longpower': Sample Size Calculations for Longitudinal Data*, 2020. URL https://cran.r-project.org/web/packages/longpower/longpower.pdf. R package version 1.0-21. [p267]

G. M. Fitzmaurice, N. M. Laird, and J. Ware. *Applied Longitudinal Analysis*. John Wiley & Sons, Hoboken, N.J., 2004. [p266]

P. Green and C. J. MacLeod. Simr: an r package for power analysis of generalized linear mixed models by simulation. *Methods in Ecology and Evolution*, 7:493–498, 2016. doi: doi:10.1111/2041-210X.12504. [p265]

X. Guo and W. D. Johnson. Sample size for experiments with multi-variate repeated measures. *J. Biopharmaceutical Statistics*, 6:155–176, 1996. [p264]

Y. Guo, H. Logan, and D. e. a. Glueck. Selecting a sample size for studies with repeated measures. *BMC Medical Research Methodology*, 13:100, 2013. doi: https://doi.org/10.1186/1471-2288-13-100. [p264]

D. Hedeker, R. D. Gibbons, and C. Waternaux. Sample size estimation for longitudinal designs with attrition: comparing time-related contrasts between two groups. *Journal of Educational and Behavioral Statistics*, 24:70–93, 1999. [p264]

K. Hemming, J. Kasza, R. Hooper, and M. Forbes, A. Taljaard. A tutorial on sample size calculation for multiple-period cluster-randomized parallel, cross-over, and stepped-wedge trials using the shiny crt calculator. *International Journal of Epidemiology*, 49(3):979–995, 2020. [p265]

N. Hu and Z. Qu. *Mixed Model Power & Size*, 2021. URL https://rcrm-power-size.shinyapps.io/rcrm_power_size/. a Shiny application accessed on 18/11/2021. [p265]

N. Hu, H. Mackey, and R. Thomas. Power and sample size for random coefficient regression models in randomized experiments with monotone missing data. *Biometrical Journal*, 2021:1–9, 2021. doi: DOI:10.1002/bimj.202000184. [p267]

K. Kleinman. *'clusterPower': Power Calculations for Cluster-Randomized and Cluster-Randomized Crossover Trials*, 2021. URL https://cran.r-project.org/web/packages/clusterPower/clusterPower.pdf. R package version 0.7.0. [p265]

F. Kloprogge and J. Tarning. *'PharmPow': Pharmacometric Power calculations for mixed study designs*, 2015. URL https://cran.r-project.org/web/packages/PharmPow/PharmPow.pdf. R package version 1.0. [p265]

N. M. Laird and J. H. Ware. Random-effects models for longitudinal data. *Biometrics*, pages 963–974, 1982. [p265]

P. Lane. Handling drop-out in longitudinal clinical trials: a comparison of the locf and mmrm approaches. *Pharmaceutical Statistics*, 7:93–106, 2008. [p267]

J. J. Lefante. The power to detect differences in average rates of change in longitudinal studies. *Statistics in Medicine*, 9:437–446, 1990. [p264]

K.-Y. Liang and S. L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73: 13–22, 1986. [p266]

A. Liu, W. J. Shih, and E. Gehan. Sample size and power determination for clustered repeated measurements. *Statistics in Medicine*, 21:1787–1801, 2002. doi: https://doi.org/10.1002/sim.1154. [p264]

G. Liu and K. Y. Liang. Sample size calculations for studies with correlated observations. *Biometrics*, 53(3):937–47, 1997. [p266, 268, 269, 272]

K. Lu, X. Luo, and P. Chen. Sample size estimation for repeated measures analysis in randomized clinical trials with missing data. *The International Journal of Biostatistics*, 4(1):Article 9, 2008. [p264, 265, 267, 268, 272]

K. Lu, D. V. Mehrotra, and G. Liu. Sample size determination for constrained longitudinal data analysis. *Statistics in Medicine*, 28:679–699, 2009. [p264]

K. J. Lui. Sample size requirement for repeated measurements in continuous data. *Statistics in Medicine*, 11:633–641, 1992. [p264]

C. Mallinckrodt, T. Sanger, S. Dubé, D. DeBrota, G. Molenberghs, R. Carroll, W. Potter, and G. Tollefson. Assessing and interpreting treatment effects in longitudinal clinical trials with missing data. *Biological Psychiatry*, 53:754–760, 2003. [p265, 267]

C. H. Mallinckrodt, W. S. Clark, and S. R. David. Accounting for dropout bias using mixed-effects models. *Journal of Biopharmaceutical Statistics*, 11:9–21, 2001. [p265, 267]

G. A. J. Martin, H. D. Nussey, J. A. Wilson, and D. Reale. Measuring individual differences in reaction norms in field and experimental studies: a power analysis of random regression models. *Methods in Ecology and Evolution*, 2:362–374, 2011. [p265]

NCSS Statistical Software. *Power Analysis & Sample Size*, 2021. URL https://www.ncss.com/software/pass/pass-documentation/. NCSS, LLC, Kaysville, Utah, USA. [p265]

nQuery. *Sample Size and Power Calculation*, 2021. URL https://www.statsols.com/nquery/sample-size-software-for-teams. 'Statsols' (Statistical Solutions Ltd), Cork, Ireland. [p265]

J. E. Overall and S. R. Doyle. Estimating sample size for repeated measurement designs. *Controlled Clinical Trials*, 15:100–123, 1994. [p264]

H. Pan, S. Liu, D. Miao, and et al. Sample size determination for mediation analysis of longitudinal data. *BMC Medical Research Methodology*, 18:32, 2018. doi: https://doi.org/10.1186/s12874-018-0473-2. [p264]

W. Qiu. *'powerMediation': Power/Sample Size Calculation for Mediation Analysis*, 2021. URL https://cran.r-project.org/web/packages/powerMediation/powerMediation.pdf. R package version 0.3.4. [p265]

J. Rochon. Sample size calculations for two-group repeated-measures experiments. *Biometrics*, 47: 1383–1398, 1991. [p264]

W. Rosen, R. Mohs, and K. Davis. Alzheimer's Disease Assessment Scale—Cognitive and Non-Cognitive sections (ADAS-Cog, ADAS Non-Cog). *Journal of Psychiatry*, 141:1356–1364, 1984. [p268]

A. Roy, D. K. Bhaumik, S. Aryal, and G. R. D. Sample size determination for hierarchical longitudinal designs with differential attrition rates. *Biometrics*, 63:699–707, 2007. doi: https://doi.org/10.1111/j.1541-0420.2007.00769.x. [p264]

A. M. Schoemann, A. J. Boulton, and S. D. Short. Determining power and sample size for simple and complex mediation models. *Social Psychological and Personality Science*, 8:379–386, 2017. [p265]

D. A. Schoenfeld. *'LPower': Calculates Power, Sample Size, or Detectable Effect for Longitudinal Analyses*, 2019. URL https://cran.r-project.org/web/packages/LPower/LPower.pdf. R package version 0.1.1. [p265]

E. F. Vonesh and M. A. Schork. Sample sizes in multivariate analysis of repeated measurements. *Biometrics*, 42:601–610, 1986. [p264]

M. W. Weiner, D. P. Veitch, P. S. Aisen, L. A. Beckett, N. J. Cairns, J. Cedarbaum, M. C. Donohue, R. C. Green, D. Harvey, C. R. Jack Jr, and others. Impact of the alzheimer's disease neuroimaging initiative, 2004 to 2014. *Alzheimer's & Dementia*, 11(7):865–884, 2015. [p265, 268]

X. Yan and X. Su. Sample size determination for clinical trials in patients with nonlinear disease progression. *Journal of Biopharmaceutical Statistics*, 16(1):91–105, 2006. doi: 10.1080/10543400500406579. [p264]

O. Yi and T. Panzarella. Estimating sample size for tests on trends across repeated measurements with missing data based on the interaction term in a mixed model. *Controlled Clinical Trials*, 23:481–496, 2002. [p265]

Z. Zhang, Y. Mai, and M. Yang. *'WebPower': Basic and Advanced Statistical Power Analysis*, 2021. URL https://webpower.psychstat.org. R package version 0.6. [p265]

Y. Zhao and S. D. Edland. Power formulas for mixed effects models with random slope and intercept comparing rate of change across groups. *International Journal of Biostatistics*, 2020:20200107, 2020. doi: https://doi.org/10.1515/ijb-2020-0107. [p266]

Samuel Iddi
Department of Statistics and Actuarial Science
University of Ghana
Legon-Accra, Ghana
https://orcid.org/0000-0002-2366-2774
siddi@ug.edu.gh; isamuel.gh@gmail.com
also
Data Measurement and Evaluation Unit
African Population and Health Research Center (APHRC)
Nairobi, Kenya
siddi@aphrc.org

Michael C Donohue
Alzheimer's Therapeutic Research Institute
Keck School of Medicine
University of Southern California
San Diego, USA
https://orcid.org/0000-0001-6026-2238
mdonohue@usc.edu

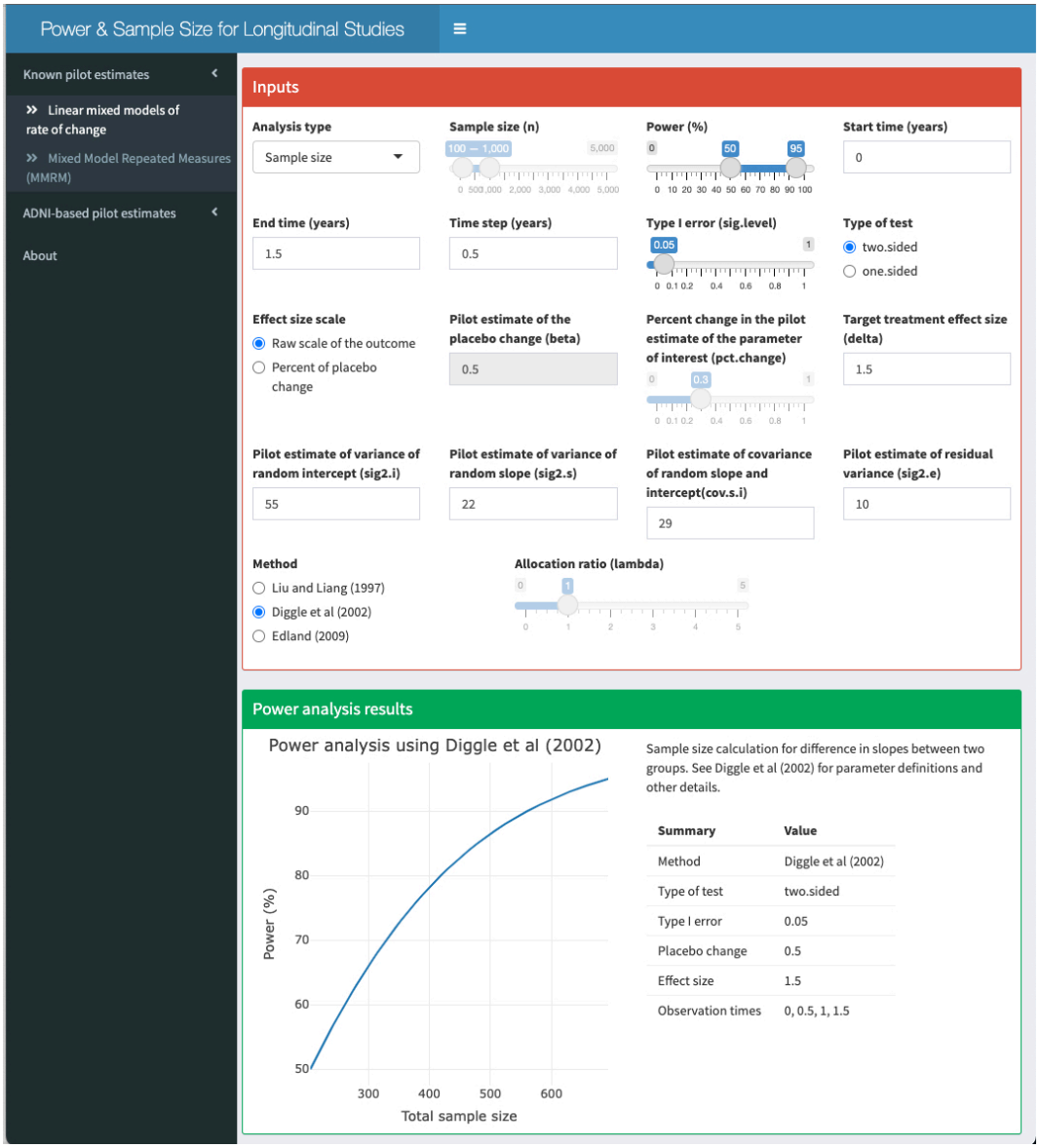


Figure 1: Screenshot of Shiny app menu for linear mixed models (LMM) of rates of change. The red box (top) denotes an area where user input of pilot estimates and study design features is required. Output, including a power curve and a summary table of study design characteristics, is displayed in the green box (bottom).

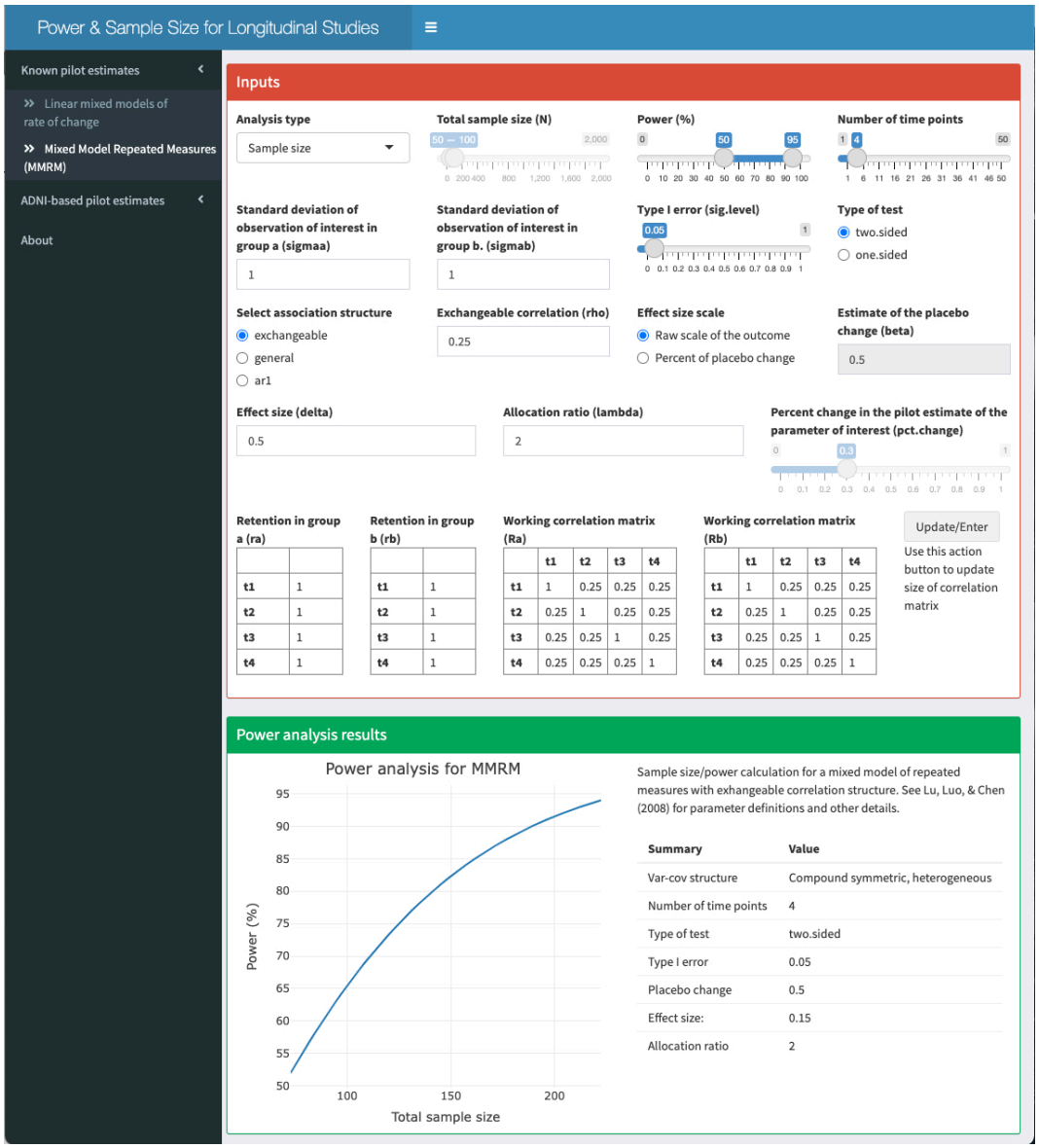


Figure 2: Screenshot of Shiny app menu for mixed models for repeated measures (MMRM). The red box (top) denotes an area where user input of pilot estimates and study design features is required. Output, including a power curve and a summary table of study design characteristics, is displayed in the green box (bottom).

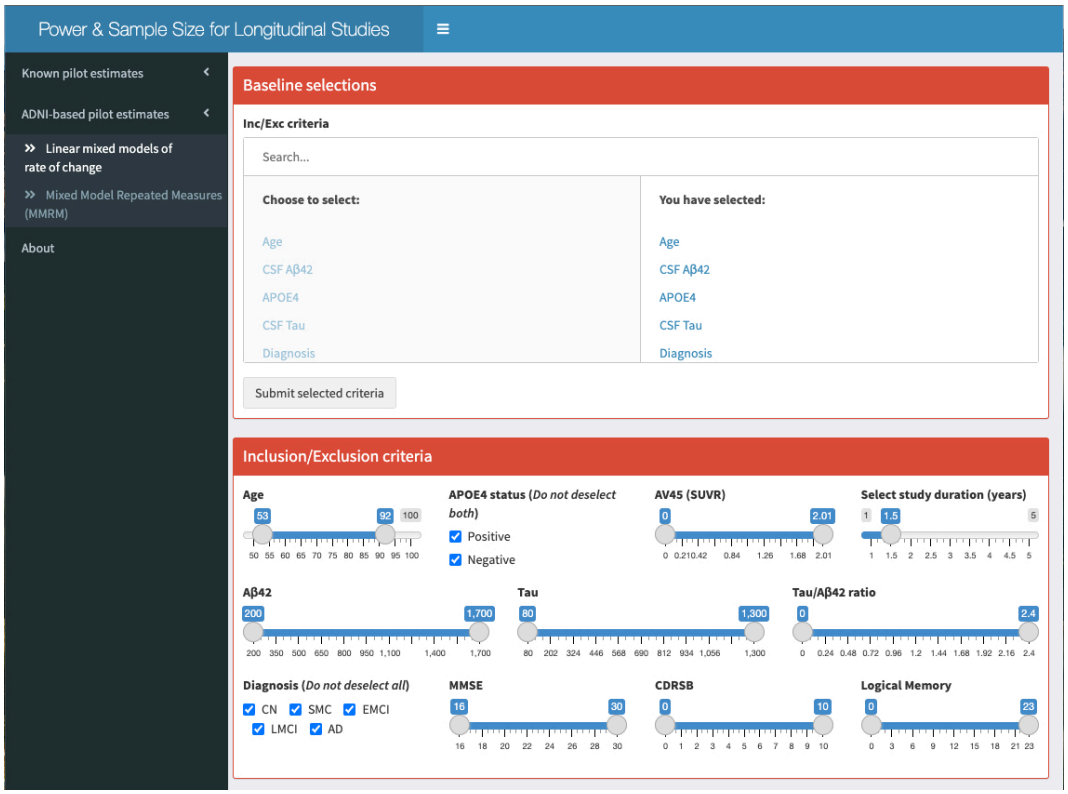


Figure 3: Screenshot of Shiny app menu for ADNI-based pilot estimate generator. The user can select variables and associated ranges for desired inclusion/exclusion criteria. The app then filters the ADNI data accordingly, summarizes the characteristics of the filtered data, fits a linear mixed model (LMM), and extracts pilot estimates required for power calculations.

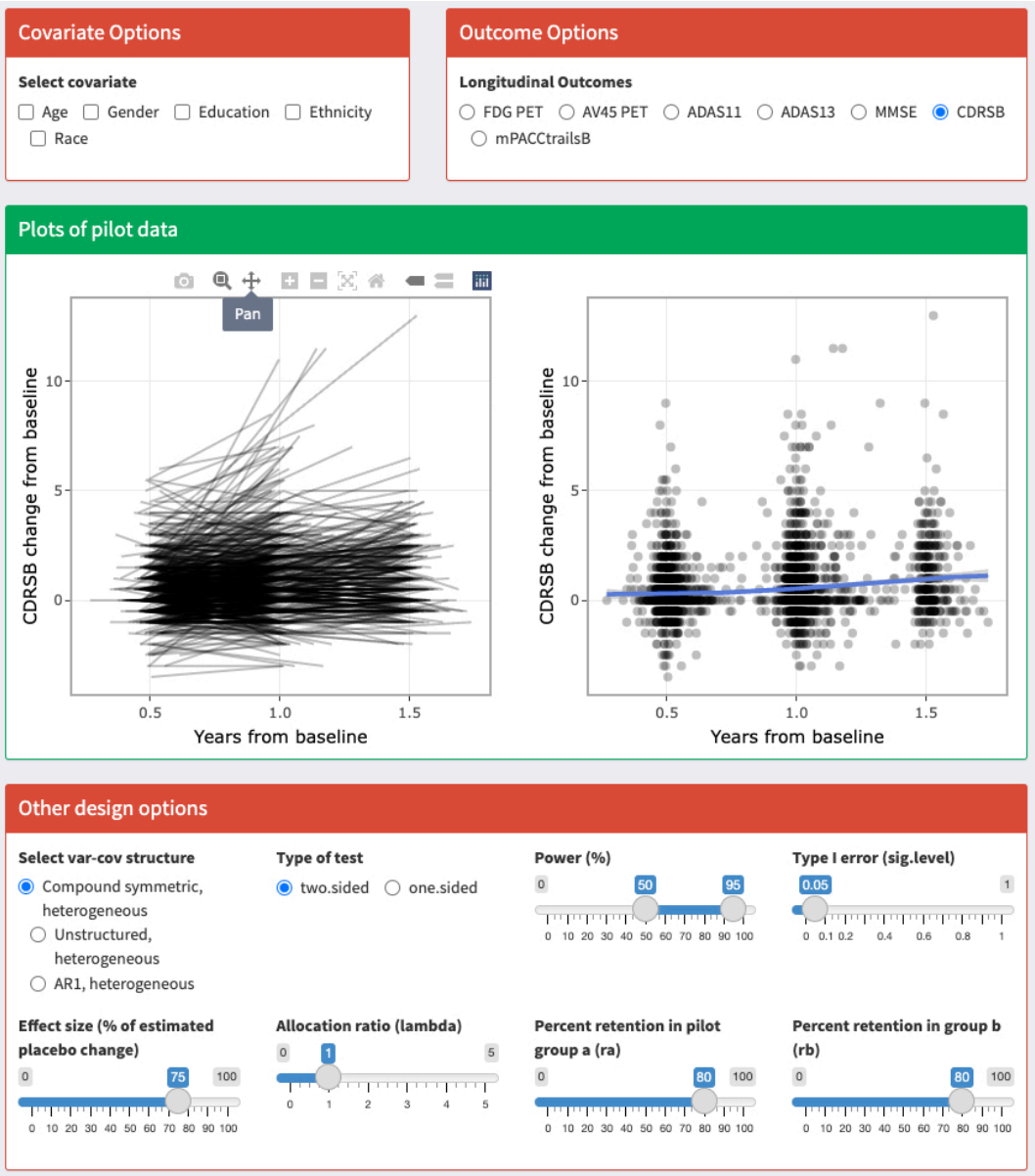


Figure 4: Screenshot of Shiny app input and output for ADNI-based pilot estimate generator. Once the user has selected variables and associated ranges for desired inclusion/exclusion criteria (Figure 3), the app summarizes the characteristics of the filtered data, including these spaghetti and dots plots (middle green box). The user must select the desired analysis model covariates and outcome (top red boxes); and other analysis model and study design features (bottom red box).

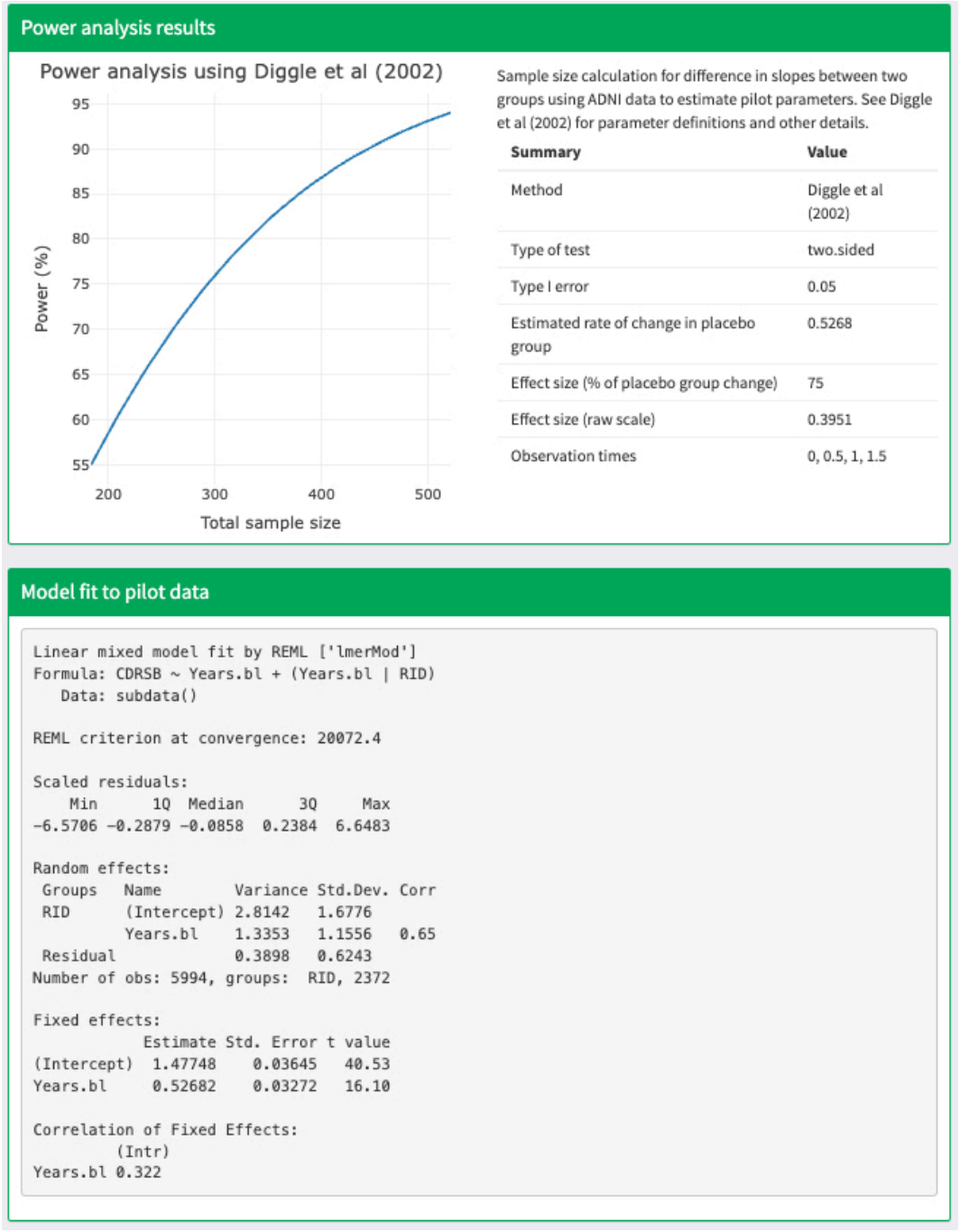


Figure 5: Screenshot of Shiny app output for ADNI-based pilot estimate generator. The bottom box shows R output summarizing a linear mixed model fit to a subset of ADNI data with desired inclusion/exclusion criteria applied. The top box shows study characteristics and a power curve derived using parameters from the fitted model.

PSweight: An R Package for Propensity Score Weighting Analysis

by Tianhui Zhou, Guangyu Tong, Fan Li, Laine E. Thomas and Fan Li

Abstract Propensity score weighting is an important tool for comparative effectiveness research. Besides the inverse probability of treatment weights (IPW), recent development has introduced a general class of balancing weights, corresponding to alternative target populations and estimands. In particular, the overlap weights (OW) lead to optimal covariate balance and estimation efficiency, and a target population of scientific and policy interest. We develop the R package **PSweight** to provide a comprehensive design and analysis platform for causal inference based on propensity score weighting. **PSweight** supports (i) a variety of balancing weights, (ii) binary and multiple treatments, (iii) simple and augmented weighting estimators, (iv) nuisance-adjusted sandwich variances, and (v) ratio estimands. **PSweight** also provides diagnostic tables and graphs for covariate balance assessment. We demonstrate the functionality of the package using a data example from the National Child Development Survey (NCDS), where we evaluate the causal effect of educational attainment on income.

1 Introduction

Propensity score is one of the most widely used causal inference methods for observational studies (Rosenbaum and Rubin, 1983). Propensity score methods include weighting, matching, stratification, regression, and mixed methods such as the augmented weighting estimators. The **PSweight** package provides a design and analysis pipeline for causal inference with propensity score weighting (Robins et al., 1994; Hirano et al., 2003; Lunceford and Davidian, 2004; Li et al., 2018). There are a number of existing R packages on propensity score weighting (see Table 1). Comparing to those, **PSweight** offers three major advantages: it incorporates (i) visualization and diagnostic tools of checking covariate overlap and balance, (ii) a general class of balancing weights, including overlap weights and inverse probability of treatment weights, and (iii) multiple treatments. More importantly, **PSweight** comprises a wide range of functionalities, whereas each of the competing packages only supports a subset of these functionalities. As such, **PSweight** is currently the most comprehensive platform for causal inference with propensity score weighting, offering analysts a one-stop shop for the design and analysis. Table 1 summarizes the key functionalities of **PSweight** in comparison to related existing R packages. We elaborate the main features of **PSweight** below.

PSweight facilitates better practices in the design stage of observational studies, an aspect that has not been sufficiently emphasized in related packages. Specifically, we provide a design module that facilitates visualizing overlap (also known as the positivity assumption) and evaluating covariate balance without access to the final outcome (Austin and Stuart, 2015). When there is limited overlap, **PSweight** allows for symmetric propensity score trimming (Crump et al., 2009; Yoshida et al., 2018) and optimal trimming (Crump et al., 2009; Yang et al., 2016) to improve the internal validity. We extend the class of balance metrics suggested in Austin and Stuart (2015) and Li et al. (2019) for binary treatments, and those in McCaffrey et al. (2013) and Li and Li (2019) for multiple treatments. In addition, the design module helps describe the weighted target population by providing the information required in the standard "Table 1" of a clinical article.

In addition to the standard inverse probability of treatment weights (IPW), **PSweight** implements the average treatment effect among the treated (Treated) weights, overlap weights (OW), matching weights (MW) and entropy weights (EW) for both binary (Li and Greene, 2013; Mao et al., 2018; Li et al., 2018; Zhou et al., 2020) and multiple treatments (Yoshida et al., 2017; Li and Li, 2019). All weights are members of the family of balancing weights (Li et al., 2018); the last three types of weights target at the subpopulation with improved overlap in the covariates between (or across) treatment groups, similar to the target population in randomized controlled trials (Thomas et al., 2020a,b). Among them, OW achieves optimal balance and estimation efficiency (Li et al., 2018, 2019). We also implement the augmented weighting estimators corresponding to each of the above weighting schemes (Mao et al., 2018). By default, **PSweight** employs parametric regression models to estimate propensity scores and potential outcomes. Nonetheless, it also allows for propensity scores to be estimated by external machine learning methods including generalized boosted regression models (McCaffrey et al., 2013) and super learner (Van der Laan et al., 2007), or importing any other propensity or outcome model estimates of interest.

To our knowledge, **PSweight** is the first R package to accommodate a variety of balancing weighting schemes with multiple treatments. Existing R packages such as **twang** (Ridgeway et al., 2020),

Table 1: Comparisons of existing R packages that implement propensity score weighting with discrete treatments. Binary treatments and additive estimands are implemented in all packages, and therefore those two columns are omitted.

	Multiple treatments	Balance diagnostics	IPW/ATT weights	OW/other weights	Ratio estimands	Augmented weighting	Nuisance-adj variance	Optimal trimming
PSweight	✓	✓	✓	✓	✓	✓	✓	✓
twang	✓	✓	✓	×	×	×	×	×
CBPS	✓	✓	✓	×	×	✓	✓	×
PSW	×	✓	✓	✓	✓	✓	✓	×
optweight	✓	×	✓	×	×	×	×	×
ATE	✓	✓	✓	×	×	×	✓	×
WeightIt	✓	×	✓	✓	×	×	×	×
causalweight	✓	×	✓	×	×	✓	×	×
sbw	×	✓	✓	×	×	×	×	×

✓ indicates that the functionality is currently implemented in the package; × indicates otherwise.
References: **twang** (Version 1.6): Ridgeway et al. (2020); **CBPS** (Version 0.21): Fong et al. (2019); **PSW** (Version 1.1-3): Mao and Li (2018); **optweight** (Version 0.2.5): Greifer (2019); **ATE** (Version 0.2.0): Haris and Chan (2015); **WeightIt** (Version 0.10.2): Greifer (2020); **causalweight** (Version 0.2.1): Bodory and Huber (2020); **sbw** (Version 1.1.1): Zubizarreta and Li (2020).

CBPS (Fong et al., 2019), **optweight** (Greifer, 2019) have also implemented weighting-based estimation with multiple treatments, but focus on IPW. The **PSW** R package (Mao and Li, 2018) implements both OW and MW and allows for nuisance-adjusted variance estimation, but it is only for binary treatments. To better assist applied researchers to perform propensity score weighting analysis, this article provides a full introduction of the **PSweight** package. In what follows, we explain the methodological foundation of **PSweight** and outline the main functions and their arguments. We further illustrate the use of these functions with a data example that studies the causal effect of educational attainment on income, and finally conclude with a short discussion.

2 Overview of propensity score weighting

Before diving into the implementation details of **PSweight**, we briefly introduce the basics of the propensity score weighting framework.

Binary treatments

Assume we have an observational study with N units. Each unit i ($i = 1, 2, \ldots, N$) has a binary treatment indicator Z_i ($Z_i = 0$ for control and $Z_i = 1$ for treated), a vector of p covariates $\boldsymbol{X}_i = (X_{1i}, \cdots, X_{pi})$. For each unit i, we assume a pair of potential outcomes $\{Y_i(1), Y_i(0)\}$ mapped to the treatment and control status, of which only the one corresponding to the observed treatment is observed, denoted by $Y_i = Z_i Y_i(1) + (1 - Z_i) Y_i(0)$; the other potential outcome is counterfactual.

Causal effects are contrasts of the potential outcomes of the same units in a *target population*, which usually is the population of a scientific interest (Thomas et al., 2020b). **PSweight** incorporates a general class of weighted average treatment effect (WATE) estimands. Specifically, assume the observed sample is drawn from a probability density $f(\boldsymbol{x})$, and let $g(\boldsymbol{x})$ denote the covariate density of the target population. The ratio $h(\boldsymbol{x}) \propto g(\boldsymbol{x})/f(\boldsymbol{x})$ is called the *tilting function*, which adjusts the distribution of the observed sample to represent the target population. Denote the conditional expectation of the potential outcome by $m_z(\boldsymbol{x}) = \mathbb{E}[Y(z)|\boldsymbol{X} = \boldsymbol{x}]$ for $z = 0, 1$. Then, we can represent the average treatment effect over the target population by a WATE estimand:

$$\tau^h = \mathbb{E}_g[Y(1) - Y(0)] = \frac{\mathbb{E}\{h(\boldsymbol{x})(m_1(\boldsymbol{x}) - m_0(\boldsymbol{x}))\}}{\mathbb{E}\{h(\boldsymbol{x})\}}. \tag{2.2.1}$$

To estimate (2.2.1), **PSweight** maintains two standard assumptions: (1) *unconfoundedness*: $\{Y(1), Y(0)\} \perp Z \mid \boldsymbol{X}$; (2) *overlap*: $0 < P(Z = 1|\boldsymbol{X}) < 1$. The propensity score is the probability of a unit being assigned to the treatment group given the covariates (Rosenbaum and Rubin, 1983): $e(\boldsymbol{x}) = P(Z = 1|\boldsymbol{X} = \boldsymbol{x})$. While assumption (1) is generally untestable and critically depends on substantive knowledge, assumption (2) can be checked visually from data by comparing the distribution of propensity scores between treatment and control groups.

For a given tilting function $h(\boldsymbol{x})$ (and correspondingly a WATE estimand τ^h), we can define the *balancing weights* (w_1, w_0) for the treated and control units: $w_1(\boldsymbol{x}) \propto h(\boldsymbol{x})/e(\boldsymbol{x})$ and $w_0(\boldsymbol{x}) \propto h(\boldsymbol{x})/\{1 - e(\boldsymbol{x})\}$. These weights balance the covariate distributions between the treated and control groups towards the target population (Li et al., 2018). **PSweight** implements the following Hájek

Table 2: Target populations, tilting functions, estimands and the corresponding balancing weights for binary treatments in **PSweight**.

Target population	Tilting function $h(\boldsymbol{x})$	Estimand	Balancing weights (w_1, w_0)
Combined	1	ATE	$\left(\frac{1}{e(\boldsymbol{x})}, \frac{1}{1-e(\boldsymbol{x})}\right)$
Treated	$e(\boldsymbol{x})$	ATT	$\left(1, \frac{e(\boldsymbol{x})}{1-e(\boldsymbol{x})}\right)$
Overlap	$e(\boldsymbol{x})(1-e(\boldsymbol{x}))$	ATO	$(1-e(\boldsymbol{x}), e(\boldsymbol{x}))$
Matching	$\xi_1(\boldsymbol{x})$	ATM	$\left(\frac{\xi_1(\boldsymbol{x})}{e(\boldsymbol{x})}, \frac{\xi_1(\boldsymbol{x})}{1-e(\boldsymbol{x})}\right)$
Entropy	$\xi_2(\boldsymbol{x})$	ATEN	$\left(\frac{\xi_2(\boldsymbol{x})}{e(\boldsymbol{x})}, \frac{\xi_2(\boldsymbol{x})}{1-e(\boldsymbol{x})}\right)$

Notes: $\xi_1(\boldsymbol{x}) = \min\{e(\boldsymbol{x}), 1-e(\boldsymbol{x})\}$ and $\xi_2(\boldsymbol{x}) = -\{e(\boldsymbol{x})\log(e(\boldsymbol{x})) + (1-e(\boldsymbol{x}))\log(1-e(\boldsymbol{x}))\}$.

estimator for WATE:

$$\hat{\tau}^h = \hat{\mu}_1^h - \hat{\mu}_0^h = \frac{\sum_{i=1}^N w_1(\boldsymbol{x}_i) Z_i Y_i}{\sum_{i=1}^N w_1(\boldsymbol{x}_i) Z_i} - \frac{\sum_{i=1}^N w_0(\boldsymbol{x}_i)(1-Z_i) Y_i}{\sum_{i=1}^N w_0(\boldsymbol{x}_i)(1-Z_i)}, \tag{2.2.2}$$

where the weights are calculated based on the propensity scores estimated from the data. Clearly, specification of $h(\boldsymbol{x})$ defines the target population and estimands. **PSweight** primarily implements the following three types of balancing weights (see Table 2 for a summary):

- *Inverse probability of treatment weights* (IPW), whose target population is the combined treatment and control group represented by the observed sample, and the target estimand is the average treatment effect among the combined population (ATE).
- *Treated weights*, whose target population is the treated group, and target estimand is the average treatment effect for the treated population (ATT). Treated weights can be viewed as a special case of IPW because it inversely weights the control group.
- *Overlap weights* (OW) (Li et al., 2018; Li and Li, 2019), whose target population is the subpopulation with the most overlap in the observed covariates between treatment and control groups . In medicine this is known as the population in clinical equipoise and is the population eligible to be enrolled in randomized clinical trials. The target estimand of OW is the average treatment effect for the overlap population (ATO).

IPW has been the dominant weighting method in the literature, but has a well-known shortcoming of being sensitive to extreme propensity scores, which induces bias and large variance in estimating treatment effects. OW addresses the conceptual and operational problems of IPW. Among all balancing weights, OW leads to the smallest asymptotic (and often finite-sample) variance of the weighting estimator (2.2.2). (Li et al., 2018, 2019). Recent simulations also show that OW provides more stable causal estimates under limited overlap (Li et al., 2019; Mao et al., 2018; Yoshida et al., 2017, 2018), and is more robust to misspecification of the propensity score model (Zhou et al., 2020).

PSweight implements two additional types of balancing weights: matching weights (MW) (Li and Greene, 2013), and entropy weights (EW) (Zhou et al., 2020). Similar to OW, MW and EW focus on target populations with substantial overlap between treatment groups. Though having similar operating characteristics, MW and EW do not possess the same theoretical optimality as OW, and are less used in practice. Therefore, we will not separately describe MW and EW hereafter.

In observational studies, propensity scores are generally unknown and need to be estimated. Therefore, propensity score analysis usually involves two steps: (1) estimating the propensity scores, and (2) estimating the causal effects based on the estimated propensity scores. In **PSweight**, the default model for estimating propensity scores with binary treatments is a logistic regression model. Spline or polynomial models can be easily incorporated by adding `bs()`, `ns()` or `poly()` terms into the model formula. **PSweight** also allows for importing propensity scores estimated from external routines, such as boosted models or super learner.

Goodness-of-fit of the propensity score model is usually assessed based on the resulting covariate balance. In the context of propensity score weighting, this is measured based on either the absolute standardized difference (ASD):

$$\text{ASD} = \left| \frac{\sum_{i=1}^N w_1(\boldsymbol{x}_i) Z_i X_{pi}}{\sum_{i=1}^N w_1(\boldsymbol{x}_i) Z_i} - \frac{\sum_{i=1}^N w_0(\boldsymbol{x}_i)(1-Z_i) X_{pi}}{\sum_{i=1}^N w_0(\boldsymbol{x}_i)(1-Z_i)} \right| \Bigg/ \sqrt{\frac{s_1^2 + s_0^2}{2}}, \tag{2.2.3}$$

or the target population standardized difference (PSD), $\max\{\text{PSD}_0, \text{PSD}_1\}$, where

$$\text{PSD}_z = \left| \frac{\sum_{i=1}^N w_z(\boldsymbol{x}_i)\mathbb{1}\{Z_i = z\}X_{pi}}{\sum_{i=1}^N w_z(\boldsymbol{x}_i)\mathbb{1}\{Z_i = z\}} - \frac{\sum_{i=1}^N h(\boldsymbol{x}_i)X_{pi}}{\sum_{i=1}^N h(\boldsymbol{x}_i)} \right| \Big/ \sqrt{\frac{s_1^2 + s_0^2}{2}}. \tag{2.2.4}$$

In (2.2.3) and (2.2.4), s_z^2 is the variance (either unweighted or weighted, depending on user specification) of the pth covariate in group z, and (w_0, w_1) are the specified balancing weights. Setting $w_0 = w_1 = 1$ corresponds to the unweighted mean differences. ASD and PSD are often displayed as column in the baseline characteristics table (known as the "Table 1") and visualized via a Love plot (also known as a forest plot) (Greifer, 2018). A rule of thumb for determining adequate balance is when ASD of all covariates is controlled within 0.1 (Austin and Stuart, 2015).

Multiple treatments

Li and Li (2019) extend the framework of balancing weights to multiple treatments. Assume that we have J ($J \geq 3$) treatment groups, and let Z_i stand for the treatment received by unit i, $Z_i \in \{1, \ldots, J\}$. We further define $D_{ij} = \mathbb{1}\{Z_i = j\}$ as a set of multinomial indicator, satisfying $\sum_{i=1}^J D_{ij} = 1$ for all j. Denote the potential outcome for unit i under treatment j as $Y_i(j)$, of which only the one corresponding to the received treatment, $Y_i = Y_i(Z_i)$, is observed. The generalized propensity score is the probability of receiving a potential treatment j given $\boldsymbol{X}$ (Imbens, 2000): $e_j(\boldsymbol{x}) = P(Z = j|\boldsymbol{X} = \boldsymbol{x})$, with the constraint that $\sum_{j=1}^J e_j(\boldsymbol{x}) = 1$.

To define the target estimand, let $m_j(\boldsymbol{x}) = \mathbb{E}[Y(j)|\boldsymbol{X} = \boldsymbol{x}]$ be the conditional expectation of the potential outcome in group j. For specified tilting function $h(\boldsymbol{x})$ and target density $g(\boldsymbol{x}) \propto f(\boldsymbol{x})h(\boldsymbol{x})$, the jth average potential outcome among the target population is

$$\mu_j^h = \mathbb{E}_g[Y(j)] = \frac{\mathbb{E}\{h(\boldsymbol{x})m_j(\boldsymbol{x})\}}{\mathbb{E}\{h(\boldsymbol{x})\}}. \tag{2.2.5}$$

Causal estimands can then be constructed in a general manner as contrasts based on μ_j^h. For example, the most commonly seen estimands in multiple treatments are the pairwise average treatment effects between groups j and j': $\tau_{j,j'}^h = \mu_j^h - \mu_{j'}^h$. This definition can be generalized to arbitrary linear contrasts. Denote $\boldsymbol{a} = (a_i, \cdots, a_J)$ as a contrast vector of length J. A general class of additive estimands is

$$\tau^h(\boldsymbol{a}) = \sum_{j=1}^J a_j \mu_j^h. \tag{2.2.6}$$

Specific choices for $\boldsymbol{a}$ with nominal and ordinal treatments can be found in Li and Li (2019). Similar to before, propensity score weighting analysis with multiple treatments rests on two assumptions: (1) *weak unconfoundedness*: $Y(j) \perp \mathbb{1}\{Z = j\}|\boldsymbol{X}$, for all j, and (2) *Overlap*: the generalized propensity score is bounded away from 0 and 1: $0 < e_j(\boldsymbol{x}) < 1$, for all j.

With multiple treatments, the tilting function $h(\boldsymbol{x})$ specifies the target population, estimand, and balancing weights. For a given $h(\boldsymbol{x})$, the balancing weights for the jth treatment group $w_j(\boldsymbol{x}) \propto h(\boldsymbol{x})/e_j(\boldsymbol{x})$. Then the Hájek estimator for μ_j^h is

$$\hat{\mu}_j^h = \frac{\sum_{i=1}^N w_j(\boldsymbol{x}_i)D_{ij}Y_i}{\sum_{i=1}^N w_j(\boldsymbol{x}_i)D_{ij}}. \tag{2.2.7}$$

Contrasts based on $\hat{\mu}_j^h$ can be obtained for any $\boldsymbol{a}$ to estimate the additive causal estimand $\tau^h(\boldsymbol{a})$. Of note, we only consider types of estimands that are transitive, and therefore the ATT estimands introduced in Lechner (2001) is not implemented. In parallel to binary treatments **PSweight** implements five types of balancing weights with multiple treatments: IPW, treated weights, OW, MW, and EW, and the corresponding target estimand of each weighting scheme is its pairwise (between each pair of treatments) counterpart in binary treatments.

Among all the weights, OW minimizes the total asymptotic variances of all pairwise comparisons, and has been shown to have the best finite-sample efficiency in estimating pairwise WATEs (Li and Li, 2019). Table 3 summarizes the target population, tilting function and balancing weight for multiple treatments that are available in **PSweight**.

To estimate the generalized propensity scores for multiple treatments, the default model in **PSweight** is a multinomial logistic model. **PSweight** also allows for externally estimated generalized propensity scores. Goodness-of-fit of the generalized propensity score model is assessed by

Table 3: Target populations, tilting functions, and the corresponding balancing weights for multiple treatments in **PSweight**.

Target population	Tilting function $h(\boldsymbol{x})$	Balancing weights $\{w_j(\boldsymbol{x}),\ j = 1, \ldots, J\}$
Combined	1	$\{1/e_j(\boldsymbol{x})\}$
Treated (j'th group)	$e_{j'}(\boldsymbol{x})$	$\left\{e_{j'}(\boldsymbol{x})/e_j(\boldsymbol{x})\right\}$
Overlap	$\{\sum_{k=1}^{J} 1/e_k(\boldsymbol{x})\}^{-1}$	$\left\{\{\sum_{k=1}^{J} 1/e_k(\boldsymbol{x})\}^{-1}/e_j(\boldsymbol{x})\right\}$
Matching	$\min_k\{e_k(\boldsymbol{x})\}$	$\{\min_k\{e_k(\boldsymbol{x})\}/e_j(\boldsymbol{x})\}$
Entropy	$-\sum_{k=1}^{J} e_k(\boldsymbol{x})\log\{e_k(\boldsymbol{x})\}$	$\left\{-\sum_{k=1}^{J} e_k(\boldsymbol{x})\log\{e_k(\boldsymbol{x})\}/e_j(\boldsymbol{x})\right\}$

the resulting covariate balance, which is measured by the pairwise versions of the ASD and PSD. The detailed formula of these metrics can be found in Li and Li (2019). A common threshold for balance is that the maximum pairwise ASD or maximum PSD is below 0.1.

Propensity score trimming

Propensity score trimming excludes units with estimated (generalized) propensity scores close to zero (or one). It is a popular approach to address the extreme weights problem of IPW. **PSweight** implements the symmetric trimming rules in Crump et al. (2009) and Yoshida et al. (2018). Operationally, we allow users to specify a single cutoff δ on the estimated generalized propensity scores, and only includes units for analysis if $\min_j\{e_j(\boldsymbol{x})\} \in [\delta, 1]$. With binary treatments, the symmetric trimming rule reduces to $e(\boldsymbol{x}) \in [\delta, 1-\delta]$. The natural restriction $\delta < 1/J$ must be satisfied due to the constraint $\sum_{j=1}^{J} e_j(\boldsymbol{x}) = 1$. To avoid specifying an arbitrary trimming threshold δ, **PSweight** also implements the optimal trimming rules of Crump et al. (2009) and Yang et al. (2016), which minimizes the (total) asymptotic variance(s) for estimating the (pairwise) ATE among the class of all trimming rules. OW can be viewed as a continuous version of trimming because it smoothly down-weigh the units with propensity scores close to 0 or 1, and thus avoids specifying a threshold.

Augmented weighting estimators

PSweight also implements augmented weighting estimators, which augment a weighting estimator by an outcome regression and improves the efficiency. With IPW, the augmented weighting estimator is known as the doubly-robust estimator (Lunceford and Davidian, 2004; Bang and Robins, 2005; Funk et al., 2011). With binary treatments, the augmented estimator with general balancing weights are discussed Hirano et al. (2003) and Mao et al. (2018). Below, we briefly outline the form of this estimator with multiple treatments. Recall the conditional mean of $Y_i(j)$ given $\boldsymbol{X}_i$ and treatment $Z_i = j$ as $m_j(\boldsymbol{x}_i) = \mathbb{E}[Y_i(j)|\boldsymbol{X}_i = \boldsymbol{x}_i] = \mathbb{E}[Y_i|\boldsymbol{X}_i = \boldsymbol{x}_i, Z_i = j]$. This conditional mean can be estimated by generalized linear models, kernel estimators, or machine learning models. **PSweight** by default employs the generalized linear models, but also allows estimated values from other routines. When $m_j(\boldsymbol{x}_i)$ is estimated by generalized linear models, **PSweight** currently accommodates three types of outcomes: continuous, binary and count outcomes (with or without an offset), using the canoncal link function.

With a pre-specified tilting function, the augmented weighting estimator for group j is

$$\hat{\mu}_j^{h,\text{aug}} = \frac{\sum_{i=1}^{N} w_j(\boldsymbol{x}_i) D_{ij}\{Y_i - m_j(\boldsymbol{x}_i)\}}{\sum_{i=1}^{N} w_j(\boldsymbol{x}_i) D_{ij}} + \frac{\sum_{i=1}^{N} h(\boldsymbol{x}_i) m_j(\boldsymbol{x}_i)}{\sum_{i=1}^{N} h(\boldsymbol{x}_i)}. \tag{2.2.8}$$

The first term of (2.2.8) is the Hájek estimator of the regression residuals, and the second term is the standardized average potential outcome (a g-formula estimator). With IPW, (2.2.8) is consistent to $\mathbb{E}[Y(j)]$ when either the propensity score model or the outcome model is correctly specified, but not necessarily both. For other balancing weights, (2.2.8) is consistent to the WATE when the propensity model is correctly specified, regardless of outcome model specification. When both models are correctly specified, (2.2.8) achieves the lower bound of the variance for regular and asymptotic linear estimators (Robins et al., 1994; Hirano et al., 2003; Mao et al., 2018).

Ratio causal estimands

With binary and count outcomes, ratio causal estimands are often of interest. Using notation from the multiple treatments as an example, once we use weighting to obtain estimates for the set of average

potential outcomes $\{\mu_j^h, j = 1, \ldots, J\}$, we can directly estimate the causal relative risk (RR) and causal odds ratio (OR), defined as

$$\tau_{j,j'}^{h,\mathrm{RR}} = \frac{\mu_j^h}{\mu_{j'}^h}, \qquad \tau_{j,j'}^{h,\mathrm{OR}} = \frac{\mu_j^h/(1-\mu_j^h)}{\mu_{j'}^h/(1-\mu_{j'}^h)}. \tag{2.2.9}$$

Here the additive estimand $\tau_{j,j'}^{h,\mathrm{RD}} = \mu_j^h - \mu_{j'}^h$ is the causal risk difference (RD). **PSweight** supports a class of ratio estimands for any given contrasts $\boldsymbol{a}$. Specifically, we define the log-RR type parameters by

$$\lambda^{h,\mathrm{RR}}(\boldsymbol{a}) = \sum_{j=1}^{J} a_j \log\left(\mu_j^h\right), \tag{2.2.10}$$

and the log-OR type parameters by

$$\lambda^{h,\mathrm{OR}}(\boldsymbol{a}) = \sum_{j=1}^{J} a_j \left\{\log\left(\mu_j^h\right) - \log\left(1-\mu_j^h\right)\right\}. \tag{2.2.11}$$

With nominal treatments, the contrast vector $\boldsymbol{a}$ can be specified to encode pairwise comparisons in the log scale (as in (2.2.10)) or in the log odds scale (as in (2.2.11)), in which case $\exp\{\lambda^{h,\mathrm{RR}}(\boldsymbol{a})\}$ and $\exp\{\lambda^{h,\mathrm{OR}}(\boldsymbol{a})\}$ become the causal RR and causal OR in (2.2.9). User-specified contrasts $\boldsymbol{a}$ can provide a variety of nonlinear estimands. For example, when $J = 3$, with $\boldsymbol{a} = (1, -2, 1)^T$ one can use **PSweight** to assess the equality of two consecutive causal RR: $H_0 : \mu_3^h/\mu_2^h = \mu_2^h/\mu_1^h$.

Variance and interval estimation

PSweight by default implements the empirical sandwich variance for propensity score weighting estimators (Lunceford and Davidian, 2004; Li et al., 2019; Mao et al., 2018) based on the M-estimation theory (Stefanski and Boos, 2002). The variance adjusted for the uncertainty in estimating the propensity score and outcome models, and are sometime referred to as the nuisance-adjusted sandwich variance. Below we illustrate the main steps with multiple treatments and general balancing weights. Write $\boldsymbol{\theta} = \left(\nu_1, \ldots, \nu_J, \eta_1, \ldots, \eta_J, \boldsymbol{\beta}^T, \boldsymbol{\alpha}^T\right)^T$ as the collection of parameters to be estimated. Then $\left\{\hat{\mu}_j^{h,\mathrm{aug}} = \hat{\nu}_j + \hat{\eta}_j : j = 1, \ldots, J\right\}$ jointly solve

$$\sum_{i=1}^{N} \Psi_i(\boldsymbol{\theta}) = \sum_{i=1}^{N} \begin{pmatrix} w_1(\boldsymbol{x}_i) D_{i1}\{Y_i - m_1(\boldsymbol{x}_i;\boldsymbol{\alpha}) - \nu_1\} \\ \vdots \\ w_J(\boldsymbol{x}_i) D_{iJ}\{Y_i - m_J(\boldsymbol{x}_i;\boldsymbol{\alpha}) - \nu_J\} \\ h(\boldsymbol{x}_i)\{m_1(\boldsymbol{x}_i;\boldsymbol{\alpha}) - \eta_1\} \\ \vdots \\ h(\boldsymbol{x}_i)\{m_J(\boldsymbol{x}_i;\boldsymbol{\alpha}) - \eta_J\} \\ S_{\boldsymbol{\beta}}(Z_i, \boldsymbol{x}_i;\boldsymbol{\beta}) \\ S_{\boldsymbol{\alpha}}(Y_i, Z_i, \boldsymbol{x}_i;\boldsymbol{\alpha}) \end{pmatrix} = \boldsymbol{0},$$

where $S_{\boldsymbol{\beta}}(Z_i, \boldsymbol{x}_i;\boldsymbol{\beta})$ and $S_{\boldsymbol{\alpha}}(Y_i, Z_i, \boldsymbol{x}_i;\boldsymbol{\alpha})$ are the score functions of the propensity score model and the outcome model. The empirical sandwich variance estimator is

$$\widehat{\mathbb{V}}(\hat{\boldsymbol{\theta}}) = \left\{\sum_{i=1}^{N} \frac{\partial}{\partial \boldsymbol{\theta}^T} \Psi_i(\hat{\boldsymbol{\theta}})\right\}^{-1} \left\{\sum_{i=1}^{N} \Psi_i(\hat{\boldsymbol{\theta}}) \Psi_i^T(\hat{\boldsymbol{\theta}})\right\} \left\{\sum_{i=1}^{N} \frac{\partial}{\partial \boldsymbol{\theta}} \Psi_i^T(\hat{\boldsymbol{\theta}})\right\}^{-1}.$$

Because $\hat{\mu}_j^{h,\mathrm{aug}} = \hat{\nu}_j + \hat{\eta}_j$, the variance of arbitrary linear contrasts based on the average potential outcomes can be easily computed by applying the Delta method to the joint variance $\widehat{\mathbb{V}}(\hat{\boldsymbol{\theta}})$. For the Hájek weighting estimators, variance is estimated by removing $S_{\boldsymbol{\alpha}}(Y_i, Z_i, \boldsymbol{x}_i;\boldsymbol{\alpha})$ as well as the components involving $m_j(\boldsymbol{x}_i;\boldsymbol{\alpha})$ in $\Psi_i(\boldsymbol{\theta})$. Finally, when propensity scores and potential outcomes are not estimated through the generalized linear model or are supplied externally, or MW are used (since the tilting function is not everywhere differentiable), **PSweight** ignores the uncertainty in estimating $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ and removes $S_{\boldsymbol{\beta}}(Z_i, \boldsymbol{x}_i;\boldsymbol{\beta})$ and $S_{\boldsymbol{\alpha}}(Y_i, Z_i, \boldsymbol{x}_i;\boldsymbol{\alpha})$ in $\Psi_i(\boldsymbol{\theta})$ in the calculation of the empirical sandwich variance. Based on the estimated variance, **PSweight** computes the associated symmetric confidence intervals and p-values via the normal approximation.

For ratio causal estimands, **PSweight** applies the logarithm transformation to improve the accuracy of the normal approximation (Agresti, 2003). For estimating the variance of causal RR, we first obtain the joint variance of $\left(\log\left(\hat{\mu}_1^{h,\mathrm{aug}}\right), \ldots \log\left(\hat{\mu}_J^{h,\mathrm{aug}}\right)\right)^T$ using the Delta method, and then estimate the

variance of $\lambda^{h,\text{RR}}(\boldsymbol{a})$. Once the symmetric confidence intervals are obtained for $\lambda^{h,\text{RR}}(\boldsymbol{a})$ using the normal approximation, we can exponentiate the upper and lower confidence limits to derive the asymmetric confidence intervals for the causal RR. Confidence intervals for the causal OR are computed similarly.

PSweight also allows using bootstrap to estimate variances, which can be much more computationally intensive than the closed-form sandwich estimator but sometimes give better finite-sample performance in small samples. By default, **PSweight** resamples $R = 50$ bootstrap replicates with replacement. For each replicate, the weighting estimator (2.2.7) or the augmented weighting estimtor (2.2.8) is implemented, providing R estimates of the J average potential outcomes (an $R \times J$ matrix). Then for any contrast vector $\boldsymbol{a} = (a_1, \cdots, a_J)^T$, **PSweight** obtains R bootstrap estimates:

$$\hat{\mathbb{T}}^h(\boldsymbol{a})_{bootstrap} = \left\{ \hat{\tau}^h(\boldsymbol{a})_1 = \sum_{j=1}^{J} a_j \hat{\mu}^h_{j,1}, \ \ldots, \hat{\tau}^h(\boldsymbol{a})_R = \sum_{j=1}^{J} a_j \hat{\mu}^h_{j,R} \right\}.$$

The sample variance of $\hat{\mathbb{T}}^h(\boldsymbol{a})_{bootstrap}$ is reported by **PSweight** as the bootstrap variance; the lower and upper 2.5% quantiles of $\hat{\mathbb{T}}^h(\boldsymbol{a})_{bootstrap}$ form the 95% bootstrap interval estimate

3 Overview of package

The **PSweight** package includes two modules tailored for design and analysis of observational studies. The design module provides diagnostics to assess the adequacy of the propensity score model and the weighted target population, prior to the use of outcome data. The analysis module provides functions to estimate the causal estimands. We briefly describe the two modules below.

Design module

PSweight offers the `SumStat()` function to visualize the distribution of the estimated propensity scores, to assess the balance of covariates under different weighting schemes, and to characterize the weighted target population. It uses the following code snippet:

```
SumStat(ps.formula, ps.estimate = NULL, trtgrp = NULL, Z = NULL, covM = NULL,
    zname = NULL,  xname = NULL, data = NULL,weight = "overlap", delta = 0,
    method = "glm", ps.control = list())
```

By default, the (generalized) propensity scores are estimated by the (multinomial) logistic regression, through the argument `ps.formula`. Alternatively, `gbm()` functions in the **gbm** package (Greenwell et al., 2019) or the `SuperLearner()` function in the **SuperLearner** package (Polley et al., 2019) can also be called by using `method = "gbm"` or `method = "SuperLearner"`. Additional parameters of those functions can be supplied through the `ps.control` argument. The argument `ps.estimate` supports estimated propensity scores from external routines. `SumStat()` produces a `SumStat` object, with estimated propensity scores, unweighted and weighted covariate means for each treatment group, balance diagnostics, and effective sample sizes (defined in (Li and Li, 2019)). We then provide a `summary.SumStat()` function, which takes the `SumStat` object and summarizes weighted covariate means by treatment groups and the between-group differences in either ASD or PSD. The default options in `weighted.var = TRUE` and `metric = "ASD"` yield ASD based on weighted standard deviations in Austin and Stuart (2015). The weighted covariate means can be used to build a baseline characteristics "Table 1" to illustrate the target population where trimming or balancing weights are applied.

```
summary(object, weighted.var = TRUE, metric = "ASD")
```

Diagnostics of propensity score models can be visualized with the `plot.SumStat()` function. It takes the `SumStat` object and produces a balance plot (`type = "balance"`) based on the ASD and PSD. A vertical dashed line can be set by the `threshold` argument, with a default value equal to 0.1. The `plot.SumStat()` function can also supply density plot (`type = "density"`), or histogram (`type = "hist"`) of the estimated propensity scores. The histogram, however, is only available for the binary treatment case. The plot function is implemented as follows:

```
plot(x, type = "balance", weighted.var = TRUE, threshold = 0.1, metric = "ASD")
```

In the design stage, propensity score trimming can be carried out with the `PStrim()` function. The trimming threshold `delta` is set to 0 by default. `PStrim()` also enables optimal trimming rules (`optimal = TRUE`) that give the most statistically efficient (pairwise) subpopulation ATE, among all

Table 4: Functions in the design module of **PSweight**.

Function	Description
`SumStat()`	Generate a `SumStat` object with information of propensity scores and weighted covariate balance
`summary.SumStat()`	Summarize the `SumStat` object and return weighted covariate means by treatment groups and weighted or unweighted between-group differences in ASD or PSD
`plot.SumStat()`	Plot the distribution of propensity scores or weighted covariate balance metrics from the `SumStat` object
`PStrim()`	Trim the data set based on estimated propensity scores

possible trimming rules. A trimmed data set along with a summary of trimmed cases will be returned by `PStrim()`. This function is given below:

```
PStrim(data, ps.formula = NULL, zname = NULL, ps.estimate = NULL, delta = 0,
    optimal = FALSE, method = "glm", ps.control = list())
```

Alternatively, trimming is also anchored in the `SumStat()` function with the `delta` argument. All functions in the design module are summarized in Table 4.

Analysis module

The analysis module of **PSweight** includes two functions: `PSweight()` and `summary.PSweight()`. The `PSweight()` function estimates the average potential outcomes in the target population, $\{\mu_j^h, j = 1, \ldots, J\}$, and the associated variance-covariance matrix. By default, the empirical sandwich variance is implemented, but bootstrap variance can be obtained with the argument `bootstrap = TRUE`). The `weight` argument can take `"IPW"`, `"treated"`, `"overlap"`, `"matching"` or `"entropy"`, corresponding to the weights introduced in Tables 2 and 3. More detailed descriptions of each input argument in the `PSweight()` function can be found in Table 5. A typical `PSweight()` code snippet looks like

```
PSweight(ps.formula, ps.estimate, trtgrp, zname, yname, data, weight = "overlap",
    delta = 0, augmentation = FALSE, bootstrap = FALSE, R = 50, out.formula = NULL,
    out.estimate = NULL, family = "gaussian", ps.method = "glm", ps.control = list(),
    out.method = "glm", out.control = list())
```

Similar to the design module, the `summary.PSweight()` function synthesizes information from the `PSweight` object for statistical inference. A typical code snippet looks like

```
summary(object, contrast, type = "DIF", CI = TRUE)
```

In the `summary.PSweight()` function, the argument `type` corresponds to the three types estimands: `type = "DIF"` is the default argument that specifies the additive causal contrasts; `type = "RR"` specifies the contrast on the log scale as in equation (2.2.10); `type = "OR"` specifies the contrast on the log odds scale as in equation (2.2.11). Confidence intervals and p-values are obtained using normal approximation and reported by the `summary.PSweight()` function. The argument `contrast` represents a contrast vector $\boldsymbol{a}$ or matrix with multiple contrast row vectors. If `contrast` is not specified, `summary.PSweight()` provides all pairwise comparisons of the average potential outcomes. By default, confidence interval is printed (`CI = TRUE`); alternatively, one can print the test statistics and p-values by `CI = FALSE`.

4 Case study with the NCDS data

We demonstrate **PSweight** in a case study that estimates the causal effect of educational attainment on hourly wage, based on the National Child Development Survey (NCDS) data. The National Child Development Survey (NCDS) is a longitudinal study on children born in the United Kingdom (UK) in 1958 [1]. NCDS collected information such as educational attainment, familial backgrounds, and socioeconomic and health well being on 17,415 individuals. We followed Battistin and Sianesi (2011) to pre-process the data and obtain a subset of 3,642 males employed in 1991 with complete educational attainment and wage information for analysis. For illustration, we use the Multiple Imputation by Chained Equations in (Buuren and Groothuis-Oudshoorn, 2010) to impute missing covariates and

[1] https://cls.ucl.ac.uk/cls-studies/1958-national-child-development-study/

Table 5: Arguments for function `PSweight()` in the analysis module of **PSweight**.

Argument	Description	Default
`ps.formula`	A symbolic description of the propensity score model.	–
`ps.estimate`	An optional matrix or data frame with externally estimated (generalized) propensity scores for each observation; can also be a vector with binary treatments.	`NULL`
`trtgrp`	An optional character defining the *treated* population for estimating (pairwise) ATT. It can also be used to specify the treatment level when only a vector of values are supplied for `ps.estimate` in the binary treatment setting.	Last value in alphabetic order
`zname`	An optional character specifying the name of the treatment variable when `ps.formula` is not provided.	`NULL`
`yname`	A character specifying name of the outcome variable in `data`.	
`weight`	A character specifying the type of weights to be used.	`"overlap"`
`delta`	Trimming threshold for (generalized) propensity scores.	0
`augmentation`	Logical value of whether augmented weighting estimators should be used.	`FALSE`
`bootstrap`	Logical value of whether bootstrap is used to estimate the standard error	`FALSE`
`R`	Number of bootstrap replicates if `bootstrap = TRUE`	50
`out.formula`	A symbolic description of the outcome model to be estimated when `augmentation = TRUE`	
`out.estimate`	An optional matrix or data frame containing externally estimated potential outcomes for each observation under each treatment level.	`NULL`
`family`	A description of the error distribution and canonical link function to be used in the outcome model if `out.formula` is provided	`"gaussian"`
`ps.method`	a character to specify the method for propensity model.	`"glm"`
`ps.control`	A list to specify additional options when `method` is set to `"gbm"` or `"SuperLearner"`.	`list()`
`out.method`	A character to specify the method for outcome model.	`"glm"`
`out.control`	A list to specify additional options when `methodout` is set to `"gbm"` or `"SuperLearner"`.	`list()`

obtain a single imputed data set for all subsequent analysis.[2] The outcome variable `wage` is log of the gross hourly wage in Pound. The treatment variable is educational attainment. For the multiple treatment case, To start with, we created `Dmult` as a treatment variable with three levels: `">=A/eq"`, `"O/eq"` and `"None"`, representing advanced qualification (1,806 individuals), intermediate qualification (941 individuals) and no qualification (895 individuals). We consider twelve pre-treatment covariates or potential confounders. The variable `white` indicates whether an individual identified himself as white race; `scht` indicates the school type they attended at age 16; `qmab` and `qmab2` are math test scores at age 7 and 11; `qvab` and `qvab2` are two reading test scores at age 7 and 11; `sib_u` stands for the number of siblings; `agepa` and `agema` are the ages of parents in year 1,974; in the same year, the employment status of mother `maemp` was also collected; `paed_u` and `maed_u` are the years of education for parents. For simplicity, we will focus on IPW and the three types of weights that improve covariate overlap: OW, MW and EW.

Estimating generalized propensity scores and balance assessment

We use `Dmult`, the three-level variable, as the treatment of interest. About one half of the population attained advanced academic qualification, there are approximately equal numbers of individuals with intermediate academic qualification or no academic qualification. To illustrate the estimation and inference for ratio estimands, we also introduce a binary outcome of wage, `wagebin`. The dichotomized wage was obtained with the cutoff of the average hourly wage of actively employed British male aged 30-39 in 1991[3]. The averaged hourly wage is 8.23, and we take $\log(8.23) \approx 2.10$ as the cutoff. Among the study participants, we observe 1610 and 2032 individuals above and below the average, and we are interested in estimating the pairwise (weighted) average treatment effect of the academic qualification for obtaining above-average hourly wage.

We specify a multinominal regression model, `ps.mult`, to estimate the generalized propensity scores.

```
ps.mult <- Dmult ~ white + maemp + as.factor(scht) + as.factor(qmab)
```

[2]Ten out of twelve pre-treatment covariates we considered have missingness. The smallest missingness proportion is 4.9% and the largest missingness proportion is 17.2%. We considered one imputed complete data set for illustrative purposes, but note that a more rigorous analysis could proceed by combining analyses from multiple imputed data sets via the Rubin's rule.

[3]https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/earningsandworkinghours/

```
    as.factor(qmab2) + as.factor(qvab) + as.factor(qvab2) + paed_u + maed_u +
    agepa + agema + sib_u + paed_u * agepa + maed_u * agema
```

Then we obtain the propensity score estimates and assess weighted covariate balance with the `SumStat()` function.

```
bal.mult <-  SumStat(ps.formula = ps.mult,
    weight = c("IPW", "overlap", "matching", "entropy"), data = NCDS)
plot(bal.mult, type = "density")
```

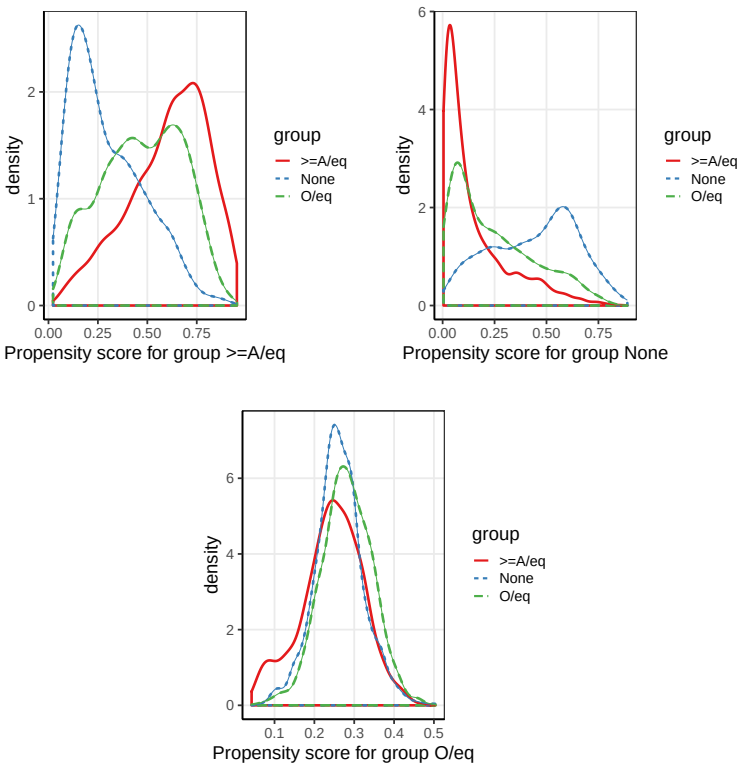


Figure 1: Density plots of estimated generalized propensity scores with respect to the three-level treatment variable `Dmult` generated by `plot.SumStat()` function in the **PSweight** package.

The distributions of generalized propensity scores are given in Figure 1 (in alphabetic order of the names of treatment groups). For the generalized propensity score to receive the advanced qualification (`">=A/eq"`) or no qualification (`"None"`), there is a mild lack of overlap due to separation of the group-specific distribution. Since `bal.mult` includes four weighting schemes, we plot the maximum pairwise ASD and assess the (weighted) covariate balance in a single Love plot.

```
plot(bal.mult, metric = "ASD")
```

The covariates are imbalanced across the three groups prior to any weighting. Although IPW can generally improve covariate balance, the maximum pairwise ASD still ocassionally exceeds the threshold 0.1 due to lack of overlap. In contrast, OW, MW and EW all emphasize the subpopulation with improved overlap and provide better balance across all covariates.

Generalized propensity score trimming

The **PSweight** package can perform trimming based on (generalized) propensity scores. As IPW does not adequately balance the covariates across the three groups in Figure 2, we explore trimming as a way to improve balance for IPW. There are two types of trimming performed by the **PSweight** package: (1) symmetric trimming that removes units with extreme (generalized propensity scores) (Crump et al., 2009; Yoshida et al., 2018) and (2) optimal trimming that provides the most efficient IPW estimator for estimating (pairwise) ATE (Crump et al., 2009; Yang et al., 2016). Specifically, the symmetric trimming is supported by both the `SumStat()` and `PSweight()` functions through the `delta` argument. Both functions refit the (generalized) propensity score model after trimming

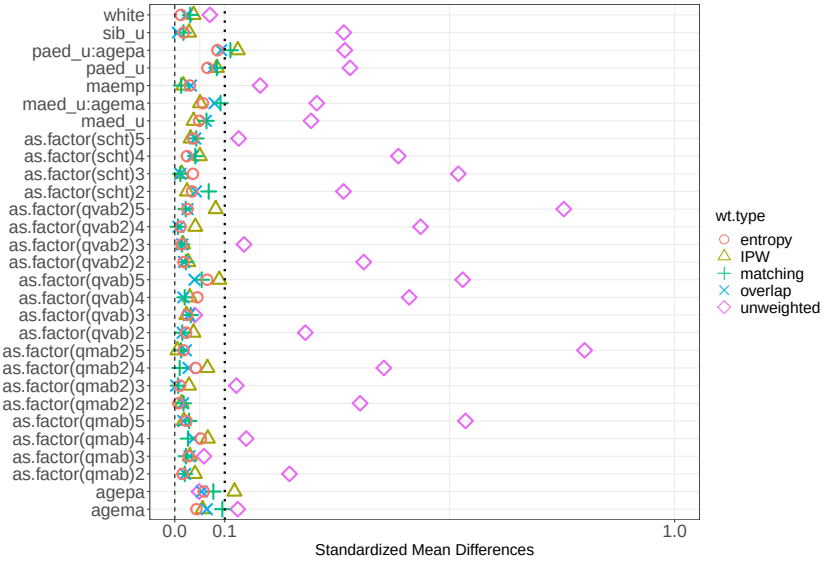


Figure 2: Love plot with the three-level treatment variable `Dmult` using the maximum pairwise ASD metric, generated by `plot.SumStat()` function in the **PSweight** package.

following the recommendations in Li et al. (2019). We also provide a stand-alone `PStrim` function that performs both symmetric trimming and optimal trimming. Following Yoshida et al. (2018), with three treatment groups, we exclude all individuals with the estimated generalized propensity scores less than $\delta = 0.067$. This threshold removes a substantial amount of individuals in the advanced qualification group (information can be pulled from the `trim` element in the `SumStat` object). As discussed in Yoshida et al. (2018), propensity trimming could improve the estimation of ATE and ATT, but barely have any effect for estimation of ATO and ATM. Evidently, Figure 3 indicates that IPW controls all pairwise ASD within 10% in the trimmed sample. Trimming had nearly no effect on the weighted balance for OW, MW and EW.

```
bal.mult.trim <- SumStat(ps.formula = ps.mult, weight = c("IPW", "overlap", "matching",
    "entropy"), data = NCDS, delta = 0.067)

bal.mult.trim

1050 cases trimmed,  2592 cases remained

trimmed result by trt group:
         >=A/eq None O/eq
trimmed     778   71  201
remained   1028  824  740

weights estimated for:  IPW overlap matching entropy

plot(bal.mult.trim,metric = "ASD")
```

Alternatively, if one does not specify the trimming threshold, the `PStrim` function supports the optimal trimming procedure that identifies the optimal threshold based on data. Example syntax is given as follows. By pulling out the summary statistics for trimming, we can see that optimal trimming excludes 27%, 9% and 2% of the individuals among those with advanced qualification, intermediate qualification and no qualification, respectively. The exclusion is more conservative compared to symmetric trimming with $\delta = 0.067$. However, the resulting covariate balance after optimal trimming is similar to Figure 3 and omitted.

```
PStrim(ps.formula = ps.mult, data = NCDS, optimal = TRUE)

         >=A/eq None O/eq
trimmed     479   21   82
remained   1327  874  859
```

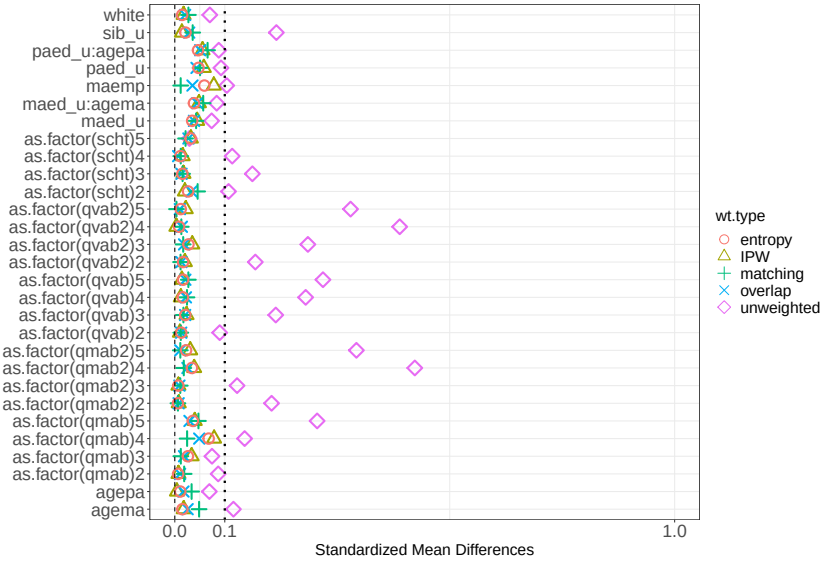


Figure 3: Love plot with the three-level treatment variable `Dmult` using the maximum pairwise ASD metric, after symmetric trimming with $\delta = 0.067$. This plot is generated by `plot.SumStat()` function in the **PSweight** package.

Estimation and inference of pairwise (weighted) average treatment effects

For illustration, we estimate the ratio estimands using the binary outcome `wagebin`. For illustration, we will only estimate the causal effects based on the data without trimming, and the analysis with the trimmed data follows the exact same steps. Based on the multinomial logistic propensity score model, we obtain the pairwise causal RR among the combined population via IPW.

```
ate.mult <- PSweight(ps.formula = ps.mult, yname = "wagebin", data = NCDS,
    weight = "IPW")}
contrasts.mult <- rbind(c(1,-1, 0), c(1, 0,-1), c(0, -1, 1))
sum.ate.mult.rr <- summary(ate.mult, type = "RR", contrast = contrasts.mult)
sum.ate.mult.rr

Closed-form inference:

Inference in log scale:
Original group value:  >=A/eq, None, O/eq

Contrast:
          >=A/eq None O/eq
Contrast 1      1   -1    0
Contrast 2      1    0   -1
Contrast 3      0   -1    1

            Estimate Std.Error       lwr     upr  Pr(>|z|)
Contrast 1  0.607027  0.115771  0.380120 0.83393 1.577e-07 ***
Contrast 2  0.459261  0.052294  0.356767 0.56176 < 2.2e-16 ***
Contrast 3  0.147766  0.121692 -0.090746 0.38628    0.2246
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

By providing the appropriate contrast matrix, we obtain all pairwise comparisons of the average potential outcomes on the log scale with the `summary.PSweight()` function, and estimate $\lambda^{h,\text{RR}}(\boldsymbol{a})$ for contrast vector $\boldsymbol{a}$. The p-values provides statistical evidence against the weak causal null H_0 : $\lambda^{h,\text{RR}}(\boldsymbol{a}) = 0$. It is found that, among the combined population, the proportion that receives an above-average hourly wage when everyone attains advanced qualification is $\exp(0.607) = 1.83$ times that when everyone attains no academic qualification. Further, the proportion that receives an above-average hourly wage when everyone attains advanced qualification is $\exp(0.459) = 1.58$ times that when everyone attains intermediate qualification. Both effects are significant at the 0.05 levels and provides strong evidence against the corresponding causal null (p-value < 0.001). However, if

everyone attains intermediate qualification, the proportion that receives an above-average hourly wage is only slightly higher compared to without qualification, with a p-value exceeding 0.05. To directly report the causal RR and its confidence intervals, we can simply exponentiate the point estimate and confidence limits provided by the `summary.PSweight()` function.

```
exp(sum.ate.mult.rr$estimates[,c(1,4,5)])
```

```
           Estimate       lwr      upr
Contrast 1 1.834968 1.4624601 2.302358
Contrast 2 1.582904 1.4287028 1.753749
Contrast 3 1.159241 0.9132496 1.471493
```

Focusing on the target population that has the most overlap in the observed covariates, we further use the OW to estimate the pairwise causal RR. OW theoretically provides the best internal validity for pairwise comparisons; Figure 3 also indicates that OW achieves better covariate balance among the overlap population. Exponentiating the results provided by the `summary.PSweight()` function, we observe each pairwise causal RR has a larger effect size among the overlap weighted population. Interestingly, among the overlap population, the proportion that receives an above-average hourly wage when everyone attains intermediate qualification becomes approximately 1.55 times that when everyone attains no academic qualification, and the associated 95% CI excludes the null. Moreover, the standard errors for the pairwise comparisons are smaller when using OW versus IPW, implying that OW analysis generally corresponds to increased power by focusing on a population with equipoise. We repeat the analysis using both MW and EW; the results are similar to OW for this analysis and therefore omitted for brevity.

```
ato.mult <- PSweight(ps.formula = ps.mult, yname = "wagebin", data = NCDS,
    weight = "overlap")
sum.ato.mult.rr <- summary(ato.mult, type = "RR", contrast = contrasts.mult)
exp(sum.ato.mult.rr$estimates[,c(1,4,5)])
```

```
           Estimate      lwr      upr
Contrast 1 2.299609 1.947140 2.715882
Contrast 2 1.527931 1.363092 1.712705
Contrast 3 1.505048 1.257180 1.801785
```

The above output suggests that among the overlap population, the causal RR for comparing advanced qualification to intermediate qualification is similar in magnitude to that for comparing intermediate qualification to no qualification. We can formally test for the equality of two consecutive causal RR based on the null hypothesis $H_0 : \mu_3^h/\mu_2^h = \mu_2^h/\mu_1^h$. Operationally, we need to specify the corresponding contrast vector `contrast = c(1,1,-2)`. The p-value for testing this null is 0.91 (output omitted for brevity), and suggests a lack of evidence against the equality of consecutive causal RR at the 0.05 level.

```
summary(ato.mult, type = "RR", contrast = c(1, 1, -2), CI = FALSE)
```

With the binary outcome `wagebin`, we can also estimate the pairwise causal OR among a specific target population. For example, using OW, the causal conclusions regarding the effectiveness due to attaining academic qualification do not change, because all three 95% confidence intervals exclude null. However, the pairwise causal OR appear larger than the pairwise causal RR. This is expected because our outcome of interest is not uncommon (Nurminen, 1995). For rare outcomes, causal OR approximates causal RR.

```
sum.ato.mult.or <- summary(ato.mult, type = "OR", contrast = contrasts.mult)
exp(sum.ato.mult.or$estimates[,c(1,4,5)])
```

```
           Estimate      lwr      upr
Contrast 1 3.586050 2.841383 4.525879
Contrast 2 2.050513 1.696916 2.477791
Contrast 3 1.748855 1.375483 2.223578
```

As a final step, we illustrate how to combine OW with outcome regression and estimate the pairwise causal RR among the overlap population. We use the same set of covariates in the binary outcome regression model.

```
 out.wagebin <- wagebin ~ white + maemp + as.factor(scht) + as.factor(qmab) +
    as.factor(qmab2) + as.factor(qvab) + as.factor(qvab2) + paed_u + maed_u +
    agepa + agema + sib_u + paed_u * agepa + maed_u * agema
```

Loading this outcome regression formula into the `PSweight()` function, and specifying `family = "binomial"` to indicate the type of outcome, we obtain the augmented overlap weighting estimates on the log RR scale. Exponentiating the point estimates and confidence limits, one reports the pairwise causal RR. The pairwise causal RR reported by the augmented OW estimator is similar to that reported by the simple OW estimator; further, the width of the confidence interval is also comparable before and after outcome augmentation, and the causal conclusions based on pairwise RR remain the same. The similarity between simple and augmented OW estimators implies that OW itself may already be efficient.

```
ato.mult.aug <- PSweight(ps.formula = ps.mult, yname = "wagebin", data = NCDS,
    augmentation = TRUE, out.formula = out.wagebin, family = "binomial")
sum.ato.mult.aug.rr <- summary(ato.mult.aug, type = "RR", contrast = contrasts.mult)
exp(sum.ato.mult.aug.rr$estimates[,c(1,4,5)])
```

```
           Estimate      lwr      upr
Contrast 1 2.310628 1.957754 2.727105
Contrast 2 1.540176 1.375066 1.725111
Contrast 3 1.500237 1.253646 1.795331
```

Using machine learning to estimate propensity scores and potential outcomes

As an alternative to the default generalized linear models, we can use more advanced machine learning models to estimate propensity scores and potential outcomes. Flexible propensity score and outcome estimation has been demonstrated to reduce bias due to model misspecification, and potentially improve covariate balance (Lee et al., 2010; Hill, 2011; McCaffrey et al., 2013). This can be achieved in **PSweight** for both balance check and constructing weighted estimator by specifying the method as the generalized boosted model (GBM) or the super learner methods. Additional model specifications for these methods can be supplied through `ps.control` and `out.control`. Machine learning models that are included in neither `gbm` nor `SuperLearner` could be estimated externally and then imported through the `ps.estimate` and `out.estimate` arguments. These two arguments broaden the utility of **PSweight** where any externally generated estimates of propensity scores and potential outcomes models can be easily incorporated.

We now illustrate the use of GBM as an alternative of the default generalized linear models. For simplicity, this illustration is based on binary education. Specifically, we created `Dany` to indicate whether one had attained any academic qualification. There are 2,399 individuals that attained academic qualification, and 1,243 individuals without any. GBM is a family of non-parametric tree-based regressions that allow for flexible non-linear relationships between predictors and outcomes (Friedman et al., 2000). The following propensity model formula is specified; the formula does not include interactions terms because boosted regression is already capable of capturing non-linear effects and interactions (McCaffrey et al., 2004). In this illustration, we use the AdaBoost (Freund and Schapire, 1997) algorithm to fit the propensity model through the control setting, `ps.control=list(distribution = "adaboost")`. We use the default values for other model parameters such as the number of trees (`n.trees = 100`), interaction depth (`interaction.depth = 1`), the minimum number of observations in the terminal nodes (`n.minobsinnode = 1`), shrinkage reduction (`shrinkage = 0.1`), and bagging fraction (`shrinkage = 0.5`). Alternative values for these parameters could also be passed through `ps.control`.

```
ps.any.gbm <- Dany ~ white + maemp + as.factor(scht) + as.factor(qmab) +
    as.factor(qmab2) + as.factor(qvab) + as.factor(qvab2) + paed_u +  maed_u+
    agepa + agema + sib_u
bal.any.gbm <-SumStat(ps.formula = ps.any.gbm, data= NCDS,  weight = "overlap",
    method = "gbm", ps.control = list(distribution = "adaboost"))
```

The balance check through `plot.SumStat()` suggests substantial improvement in covariate balance with SMD of all covariates below 0.1 after weighting. After assessing balance and confirming the adequacy of the propensity score model, we further fit the outcome model using GBM with the default logistic regression and parameters. In the `PSweight()` function, we can specify both `ps.method = "gbm"` and `out.method = "gbm"` and leave the `out.control` argument as default. The detailed code and summary of the output is in below. Here we redefine the propensity score model without interaction terms because GBM considers interactions between covariates by default. The results using GBM, in this example, are very similar to those using generalized linear models (results omitted).

```
out.wage.gbm <- wage ~ white + maemp + as.factor(scht) + as.factor(qmab) +
    as.factor(qmab2) + as.factor(qvab) + as.factor(qvab2) + paed_u +
```

```
    maed_u + agepa + agema + sib_u
ato.any.aug.gbm <- PSweight(ps.formula = ps.any.gbm, yname = "wagebin",
    data = NCDS,  augmentation = TRUE, out.formula = out.wage.gbm,
    ps.method = "gbm", ps.control = list(distribution = "adaboost"),
    out.method = "gbm")
summary(ato.any.aug.gbm, CI = FALSE)

Closed-form inference:

Original group value:  0, 1

Contrast:
           0 1
Contrast 1 -1 1

          Estimate Std.Error z value  Pr(>|z|)
Contrast 1 0.186908  0.018609  10.044 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5 Summary

Propensity score weighting is an important tool for causal inference and comparative effectiveness research. This paper introduces the **PSweight** package and demonstrates its functionality with the NCDS data example in the context of binary and multiple treatment groups. In addition to providing easy-to-read balance statistics and plots to aid the design of observational studies, the **PSweight** offers point and variance estimation with a variety of weighting schemes for the (weighted) average treatment effects on both the additive and ratio scales. These weighting schemes include the optimal overlap weights recently introduced in Li et al. (2018) and Li and Li (2019), and could help generate valid causal comparative effectiveness evidence among the population at equipoise.

Although propensity score weighting has been largely developed in observational studies, it is also an important tool for covariate adjustment in randomized controlled trials (RCTs). Williamson et al. (2014) showed that IPW can reduce the variance of the unadjusted difference-in-means treatment effect estimator in RCTs, and Shen et al. (2014) proved that the IPW estimator is semiparametric efficient and asymptotically equivalent to the analysis of covariance (ANCOVA) estimator (Tsiatis et al., 2008). Zeng et al. (2020) generalized these results of IPW to the family of balancing weights. Operationally, there is no difference in implementing propensity score weighting between RCTs and observational studies. Therefore, **PSweight** is directly applicable to perform covariate-adjusted analysis in RCTs.

The **PSweight** package is under continuing development to include other useful components for propensity score weighting analysis. Specifically, future versions of **PSweight** will include components to enable pre-specified subgroup analysis with balancing weights and flexible variable selection tools (Yang et al., 2021). We are also studying overlap weighting estimators with time-to-event outcomes and complex survey designs. Those new features are being actively developed concurrently with our extensions to the methodology.

6 Acknowledgement

Tianhui Zhou and Guangyu Tong contributed equally to this work and are considered co-first authors. The authors would like to acknowledge the support of the Patient-Centered Outcomes Research Institute (PCORI) contract ME-2018C2-13289, and the NCDS replication data published on Harvard Dataverse (https://dataverse.harvard.edu/) (Battistin and Sianesi, 2012), which provides a coded data set for our illustrative analysis.

Bibliography

A. Agresti. *Categorical Data Analysis*, volume 482. John Wiley & Sons, 2003. doi: 10.1080/02664763.2013.854979. [p287]

P. C. Austin and E. A. Stuart. Moving towards best practice when using inverse probability of treatment weighting (iptw) using the propensity score to estimate causal treatment effects in observational studies. *Statistics in Medicine*, 34(28):3661–3679, 2015. doi: 10.1002/sim.6607. [p282, 285, 288]

H. Bang and J. M. Robins. Doubly robust estimation in missing data and causal inference models. *Biometrics*, 61(4):962–973, 2005. doi: 10.1111/j.1541-0420.2005.00377.x. [p286]

E. Battistin and B. Sianesi. Misclassified treatment status and treatment effects: An application to returns to education in the United Kingdom. *Review of Economics and Statistics*, 93(2):495–509, 2011. doi: 10.1162/REST_a_00175. [p289]

E. Battistin and B. Sianesi. Replication data for: Misclassified treatment status and treatment effects: An application to returns to education in the United Kingdom, 2012. URL https://doi.org/10.7910/DVN/EPCYUL. [p296]

H. Bodory and M. Huber. **causalweight**: *Causal Inference Based on Inverse Probability Weighting, Doubly Robust Estimation, and Double Machine Learning*, 2020. URL https://CRAN.R-project.org/package=causalweight. R package version 0.2.1. [p283]

S. v. Buuren and K. Groothuis-Oudshoorn. **mice**: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, pages 1–68, 2010. doi: 10.18637/jss.v045.i03. [p289]

R. K. Crump, V. J. Hotz, G. W. Imbens, and O. A. Mitnik. Dealing with limited overlap in estimation of average treatment effects. *Biometrika*, 96(1):187–199, 2009. doi: 10.1093/biomet/asn055. [p282, 286, 291]

C. Fong, M. Ratkovic, and K. Imai. **CBPS**: *Covariate Balancing Propensity Score*, 2019. URL https://CRAN.R-project.org/package=CBPS. R package version 0.21. [p283]

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. doi: 10.1006/jcss.1997.1504. [p295]

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000. doi: 10.1214/aos/1016218223. [p295]

M. J. Funk, D. Westreich, C. Wiesen, T. Stürmer, M. A. Brookhart, and M. Davidian. Doubly robust estimation of causal effects. *American Journal of Epidemiology*, 173(7):761–767, 2011. doi: 10.1093/aje/kwq439. [p286]

B. Greenwell, B. Boehmke, J. Cunningham, and G. Developers. **gbm**: *Generalized Boosted Regression Models*, 2019. URL https://CRAN.R-project.org/package=gbm. R package version 2.1.5. [p288]

N. Greifer. Covariate balance tables and plots: A guide to the cobalt package. Technical report, Johns Hopkins University, 2018. [p285]

N. Greifer. **optweight**: *Targeted Stable Balancing Weights Using Optimization*, 2019. URL https://CRAN.R-project.org/package=optweight. R package version 0.2.5. [p283]

N. Greifer. **WeightIt**: *Weighting for Covariate Balance in Observational Studies*, 2020. URL https://CRAN.R-project.org/package=WeightIt. R package version 0.10.2. [p283]

A. Haris and G. Chan. **ATE**: *Inference for Average Treatment Effects using Covariate Balancing*, 2015. URL https://CRAN.R-project.org/package=ATE. R package version 0.2.0. [p283]

J. L. Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011. doi: 10.1198/jcgs.2010.08162. [p295]

K. Hirano, G. Imbens, and G. Ridder. Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica*, 71:1161–1189, 2003. doi: 10.3386/t0251. [p282, 286]

G. W. Imbens. The role of the propensity score in estimating dose-response functions. *Biometrika*, 87(3):706–710, 2000. doi: 10.1093/biomet/87.3.706. [p285]

M. Lechner. Identification and estimation of causal effects of multiple treatments under the conditional independence assumption. In *Econometric Evaluation of Labour Market Policies*, pages 43–58. Springer, 2001. doi: 10.2139/ssrn.177089. [p285]

B. K. Lee, J. Lessler, and E. A. Stuart. Improving propensity score weighting using machine learning. *Statistics in Medicine*, 29(3):337–346, 2010. doi: 10.1002/sim.3782. [p295]

F. Li and F. Li. Propensity score weighting for causal inference with multiple treatments. *The Annals of Applied Statistics*, 13(4):2389–2415, 2019. doi: 10.1214/19-AOAS1282. [p282, 284, 285, 286, 288, 296]

F. Li, K. L. Morgan, and A. M. Zaslavsky. Balancing covariates via propensity score weighting. *Journal of the American Statistical Association*, 113(521):390–400, 2018. doi: 10.1080/01621459.2016.1260466. [p282, 283, 284, 296]

F. Li, L. E. Thomas, and F. Li. Addressing extreme propensity scores via the overlap weights. *American Journal of Epidemiology*, 1(188):250–257, 2019. doi: 10.1093/aje/kwy201. [p282, 284, 287, 292]

L. Li and T. Greene. A weighting analogue to pair matching in propensity score analysis. *International Journal of Biostatistics*, 9(2):1–20, 2013. doi: 10.1515/ijb-2012-0030. [p282, 284]

J. K. Lunceford and M. Davidian. Stratification and weighting via the propensity score in estimation of causal treatment effects: A comparative study. *Statistics in Medicine*, 23(19):2937–2960, 2004. doi: 10.1002/sim.1903. [p282, 286, 287]

H. Mao and L. Li. **PSW**: *Propensity Score Weighting Methods for Dichotomous Treatments*, 2018. URL https://CRAN.R-project.org/package=PSW. R package version 1.1-3. [p283]

H. Mao, L. Li, and T. Greene. Propensity score weighting analysis and treatment effect discovery. *Statistical Methods in Medical Research*, page In press, 2018. doi: 10.1177/0962280218781171. [p282, 284, 286, 287]

D. F. McCaffrey, G. Ridgeway, and A. Morral. Propensity score estimation with boosted regression for evaluating causal effects in observational studies. *Psychological Methods*, 9(4):403–425, 2004. doi: 10.1037/1082-989X.9.4.403. [p295]

D. F. McCaffrey, B. A. Griffin, D. Almirall, M. E. Slaughter, R. Ramchand, and L. F. Burgette. A tutorial on propensity score estimation for multiple treatments using generalized boosted models. *Statistics in Medicine*, 32(19):3388–3414, 2013. doi: 10.1002/sim.5753. [p282, 295]

M. Nurminen. To use or not to use the odds ratio in epidemiologic analyses? *European Journal of Epidemiology*, 11(4):365–371, 1995. doi: 10.1007/BF01721219. [p294]

E. Polley, E. LeDell, C. Kennedy, and M. van der Laan. **SuperLearner**: *Super Learner Prediction*, 2019. URL https://CRAN.R-project.org/package=SuperLearner. R package version 2.0-26. [p288]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016. URL https://www.R-project.org/. ISBN 3-900051-07-0. [p]

G. Ridgeway, D. McCaffrey, A. Morral, B. A. Griffin, L. Burgette, and M. Cefalu. **twang**: *Toolkit for Weighting and Analysis of Nonequivalent Groups*, 2020. URL https://CRAN.R-project.org/package=twang. R package version 1.6. [p282, 283]

J. M. Robins, A. Rotnitzky, and L. P. Zhao. Estimation of regression-coefficients when some regressors are not always observed. *Journal of the American Statistical Association*, 89(427):846–866, 1994. doi: 10.2307/2290910. [p282, 286]

P. R. Rosenbaum and D. B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983. doi: 10.1093/biomet/70.1.41. [p282, 283]

C. Shen, X. Li, and L. Li. Inverse probability weighting for covariate adjustment in randomized studies. *Statistics in Medicine*, 33(4):555–568, 2014. doi: 10.1186/s12874-020-00947-7. [p296]

L. A. Stefanski and D. D. Boos. The calculus of m-estimation. *American Statistician*, 56(1):29–38, 2002. doi: 10.1198/000313002753631330. [p287]

L. E. Thomas, F. Li, and M. J. Pencina. Overlap weighting: A propensity score method that mimics attributes of a randomized clinical trial. *Journal of the American Medical Association*, 323(23):2417–2418, 2020a. doi: 10.1001/jama.2020.7819. [p282]

L. E. Thomas, F. Li, and M. J. Pencina. Using propensity score methods to create target populations in observational clinical research. *Journal of the American Medical Association*, 323(5):466–467, 2020b. doi: 10.1001/jama.2019.21558. [p282, 283]

A. A. Tsiatis, M. Davidian, M. Zhang, and X. Lu. Covariate adjustment for two-sample treatment comparisons in randomized clinical trials: A principled yet flexible approach. *Statistics in Medicine*, 27(23):4658–4677, 2008. doi: 10.1002/sim.3113. [p296]

M. J. Van der Laan, E. C. Polley, and A. E. Hubbard. Super Learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007. doi: 10.2202/1544-6115.1309. [p282]

E. J. Williamson, A. Forbes, and I. R. White. Variance reduction in randomised trials by inverse probability weighting using the propensity score. *Statistics in Medicine*, 33(5):721–737, 2014. doi: 10.1002/sim.5991. [p296]

S. Yang, G. W. Imbens, Z. Cui, D. E. Faries, and Z. Kadziola. Propensity score matching and subclassification in observational studies with multi-level treatments. *Biometrics*, 72(4):1055–1065, 2016. doi: 10.1111/biom.12505. [p282, 286, 291]

S. Yang, E. Lorenzi, G. Papadogeorgou, D. M. Wojdyla, F. Li, and L. E. Thomas. Propensity score weighting for causal subgroup analysis. *Statistics in Medicine*, 2021. [p296]

K. Yoshida, S. Hernández-Díaz, D. H. Solomon, J. W. Jackson, J. J. Gagne, R. J. Glynn, and J. M. Franklin. Matching weights to simultaneously compare three treatment groups comparison to three-way matching. *Epidemiology*, 28(3):387–395, 2017. doi: 10.1097/EDE.0000000000000627. [p282, 284]

K. Yoshida, D. Solomon, S. Haneuse, S. Kim, E. Patorno, S. Tedeschi, H. Lyu, T. Franklin, J.M.and Stürmer, S. Hernández-Díaz, and R. Glynn. Multinomial extension of propensity Score trimming methods: A simulation study. *American Journal of Epidemiology*, 183(3):609–616, 2018. doi: 10.1093/aje/kwy263. [p282, 284, 286, 291, 292]

S. Zeng, F. Li, R. Wang, and F. Li. Propensity Score weighting for covariate adjustment in randomized clinical trials. *Statistics in Medicine*, 40(4):842–858, 2020. [p296]

Y. Zhou, R. A. Matsouaka, and L. Thomas. Propensity core weighting under limited overlap and model misspecification. *Statistical Methods in Medical Research*, 29(12):3721–3756, 2020. doi: 10.1177/0962280220940334. [p282, 284]

J. R. Zubizarreta and Y. Li. **sbw***: Stable Balancing Weights for Causal Inference and Estimation with Incomplete Outcome Data*, 2020. URL https://CRAN.R-project.org/package=sbw. R package version 1.1.1. [p283]

Tianhui Zhou
Department of Biostatistics and Bioinformatics
Duke University School of Medicine
2424 Erwin Road, Suite 1105
Durham, NC 27705, United States of America
E-mail: tianhui.zhou@duke.edu

Guangyu Tong
Department of Biostatistics
Yale University School of Public Health
300 George St, Suite 511
New Haven, CT 06510, United States of America
E-mail: guangyu.tong@yale.edu
⋆*T. Zhou and G. Tong contributed equally to this work and are considered co-first authors.*

Fan Li
Department of Statistical Science
Duke University
122 Old Chemistry Building
Durham, NC 27705, United States of America
E-mail: fl35@duke.edu

Laine E. Thomas
Department of Biostatistics and Bioinformatics
Duke University School of Medicine
2424 Erwin Road, Suite 1105
Durham, NC 27705, United States of America
E-mail: laine.thomas@duke.edu

Fan Li
Department of Biostatistics
Yale University School of Public Health
135 College St, Suite 200
New Haven, CT 06510, United States of America
E-mail: fan.f.li@yale.edu

RFpredInterval: An R Package for Prediction Intervals with Random Forests and Boosted Forests

by Cansu Alakuş, Denis Larocque and Aurélie Labbe

Abstract Like many predictive models, random forests provide point predictions for new observations. Besides the point prediction, it is important to quantify the uncertainty in the prediction. Prediction intervals provide information about the reliability of the point predictions. We have developed a comprehensive R package, **RFpredInterval**, that integrates 16 methods to build prediction intervals with random forests and boosted forests. The set of methods implemented in the package includes a new method to build prediction intervals with boosted forests (PIBF) and 15 method variations to produce prediction intervals with random forests, as proposed by Roy and Larocque (2020). We perform an extensive simulation study and apply real data analyses to compare the performance of the proposed method to ten existing methods for building prediction intervals with random forests. The results show that the proposed method is very competitive and, globally, outperforms competing methods.

1 Introduction

Predictive modelling is the general concept of building a model that describes how a group of covariates can be used to predict a response variable. The objective is to predict the unknown responses of observations given their covariates. For example, predictive models could be used to predict the sale price of houses given house characteristics (De Cock, 2011). In its simplest form, a predictive model aims to provide a point prediction for a new observation. However, a point prediction does not contain information about its precision that can tell us how close to the true response we can expect the prediction to be, which is often important in decision-making context. Hence, although the point prediction is often the main goal of predictive analysis, assessing its reliability is equally important, and this can be achieved with a prediction interval (PI). A PI contains a set of likely values for the true response with an associated level of confidence, usually, 90% or 95%. Given that shorter PIs are more informative, developing predictive models that can produce shorter PIs along with the point predictions is crucial in assessing and quantifying the prediction error. In real-world applications, knowing the prediction error alongside the point prediction increases the practical value of the prediction.

Regression analysis is a form of predictive modelling technique that examines the relationship between a response variable and a group of covariates. In this paper, we consider a general regression model

$$Y = g(X) + \epsilon \tag{1}$$

where Y is a univariate continuous response variable, X is a p-dimensional vector of predictors, and ϵ is an error term. We assume $g(.)$ is an unknown smooth function $\Re^p \to \Re$ and $E[Y|X = x] = g(X)$. A confidence interval of the prediction is a range likely to contain the location of the response variable's true population mean. However, a prediction interval for a new observation is wider than its corresponding confidence interval and provides a range likely to contain this new observation's response value.

In the past decade, random forests have increased in popularity and provide an efficient way to generate point predictions for model (1). A random forest is an ensemble method composed of many decision trees, which can be described with a simple algorithm (Breiman, 2001). For each tree $b = \{1, ..., B\}$, a bootstrap sample of observations is drawn and a fully grown tree is built such that a set of predictors is randomly selected at each node and the best split is selected among all possible splits with those predictors only. The random forest prediction for a new observation is the average of the B trees

$$\hat{y}_{new} = \frac{1}{B}\sum_{b=1}^{B} \hat{y}^b_{new}$$

where $\hat{y}^b_{new}$ is the tree prediction for the new observation in the bth tree, i.e. the average of observations in the terminal node corresponding to the new observation. Besides this traditional description, the modern view also considers random forests as data-driven weight generators (Hothorn et al., 2004; Lin and Jeon, 2006; Moradian et al., 2017, 2019; Athey et al., 2019; Roy and Larocque, 2020; Tabib and Larocque, 2020; Alakuş et al., 2021).

Although random forests limit over-fitting by combining many trees, which reduces the variance of the estimator, final predictions can be biased (Mentch and Hooker, 2016; Wager and Athey, 2018). Since each tree is built under the same random process, all trees focus on the same part of the response signal, usually the strongest. Therefore, some parts of the response signal may be left untargeted, which could result in biased point predictions. Wager and Athey (2018) provide bounds for the extent of the bias of random forests under some assumptions about the tree growing process. Following their work, Ghosal and Hooker (2021) proposed a bias correction method in a regression framework called *one-step boosted forest*, which is introduced in Breiman (2001) and Zhang and Lu (2012). The main idea of the proposed method is to sum the predictions of two random forests, where the first is a regression forest fitted on the target data set and the second is fitted on the out-of-bag residuals of the former. Empirical studies show that this method provides a significant bias reduction when compared to a simple random forest.

The current paper proposes an R package providing, among other features, an extension of the one-step boosted forest method described above (Ghosal and Hooker, 2021). The literature on prediction intervals for random forests consists mostly of recent studies. The first method is the Quantile Regression Forests (QRF) method proposed by Meinshausen (2006). The aim of QRF is to estimate conditional quantiles of the response variable, instead of conditional means, using an estimated cumulative distribution function obtained with the nearest neighbour forest weights introduced by Hothorn et al. (2004). Prediction intervals can be built directly from the estimated conditional quantiles. The method is implemented in the CRAN package **quantregForest** (Meinshausen, 2017).

In a more recent study, Athey et al. (2019) proposed Generalized Random Forests (GRF), a very general framework to estimate any quantity, such as conditional means, quantiles or average partial effects, identified by local moment equations. Trees are grown with splitting rules designed to maximize heterogeneity with respect to the quantity of interest. Quantile regression forest is one of the applications of GRF. Similar to the QRF, the GRF method uses the neighbourhood information from different trees to compute a weighted set of neighbours for each test point. Unlike QRF, which grows trees with the least-squares criterion, GRF uses a splitting rule designed to capture heterogeneity in conditional quantiles. An implementation of quantile regression forest with GRF is available in the function `quantile_forest` of the CRAN package **grf** (Tibshirani et al., 2021).

Vovk et al. (2005, 2009) introduced a general distribution-free conformal prediction interval framework. Any predictive model, including random forests, can be used within the proposed methodology. The idea is to use an augmented data set that includes the new observation to be predicted to fit the model, and apply a set of hypothesis tests to provide an error bound around the point prediction for the new observation. Although this method does not require any distribution assumptions, it is computationally intensive. Lei et al. (2018) proposed a variant of this method, called Split Conformal (SC) prediction, which splits the data into two subsets, one to fit the model, and one to compute the quantiles of the residual distribution. We note that, while the original full conformal prediction interval framework produces shorter intervals, SC is computationally more efficient. The R package **conformalInference** (Tibshirani, 2019), available on GitHub, implements this method.

Roy and Larocque (2020) proposed 20 distinct variations of methods to improve the performance of prediction intervals with random forests. These approaches differ according to 1) the method used to build the forest and 2) the method used to build the prediction interval. Four methods can be used to build the forest: three from the classification and regression tree (CART) paradigm (Breiman and Breiman, 1984) and the transformation forest method (TRF) proposed by Hothorn and Zeileis (2021). Within the CART paradigm, in addition to the default least-squares (LS) splitting criterion, two alternative splitting criteria, L_1 and shortest prediction interval (SPI), are considered. Prediction intervals are built using the Bag of Observations for Prediction (BOP), which is the set of nearest neighbour observations previously used in Moradian et al. (2017, 2019). In addition to the type of forest chosen, there are also five methods to build prediction intervals: the classical method (LM), the quantile method (Quant), the shortest prediction interval (SPI), the highest density region (HDR), and the contiguous HDR (CHDR). LM is computed based on an intercept-only linear model using the BOP as the sample, and produces a symmetric PI around the point prediction. The quantile method, similar to the QRF method, is based on the quantiles of the BOP. SPI corresponds to the shortest interval among the intervals that contain at least $(1-\alpha)$ 100% of the observations in the BOP. As an alternative to SPI, HDR is the smallest region in the BOP, with the desired coverage $(1-\alpha)$. Note that HDR is not necessarily a single interval. If the distribution is multimodal, it can be formed by multiple intervals. Finally, CHDR is a way to obtain a single prediction interval from HDR intervals by building an interval with the minimum and maximum bounds of the HDR intervals.

Zhang et al. (2020) proposed a forest-based prediction interval method, called Out-of-Bag (OOB) prediction intervals, to estimate prediction intervals using the empirical quantiles of the out-of-bag prediction errors. The method assumes that OOB prediction errors are identically distributed and that their distribution can be well approximated by the out-of-bag prediction errors obtained from all training observations. The resulting prediction intervals have the same width for all test observations.

The method is implemented in the CRAN package **rfinterval** (Zhang, 2019).

Lu and Hardin (2021) proposed a very interesting and useful method to estimate the conditional prediction error distribution of a random forest. The main idea of the proposed method is to use a random forest to compute the out-of-bag residuals for the training observations and to form a set of out-of-bag neighbours for each test point. Then, the conditional prediction error distribution for each test point is determined with the out-of-bag residuals in the neighbourhood. Estimating the prediction error distribution enables the estimation of conditional quantiles, conditional biases and conditional mean squared prediction errors. The prediction interval for a test point x, defined by $\widehat{PI}_\alpha(x)$, is formed by adding the $\alpha/2$ and $1-\alpha/2$ quantiles of the conditional prediction error distribution to the random forest point prediction. The estimators are implemented in the CRAN package **forestError** (Lu and Hardin, 2020).

Note that the conformal inference, OOB approach of Zhang et al. (2020) and the $\widehat{PI}_\alpha(x)$ method of Lu and Hardin (2021) all use the prediction errors to build the prediction intervals. Instead of using the training responses directly to estimate quantiles, using prediction errors provides a better predictive power. However, unlike conformal inference and the OOB approach, the $\widehat{PI}_\alpha(x)$ method uses the nearest neighbour observations to estimate the prediction error distribution. This idea is very similar to the BOP idea (Roy and Larocque, 2020), but instead of using in-bag observations, Lu and Hardin (2021) use out-of-bag observations to form the neighbourhoods. This approach allows the local information for the test observations to be extracted.

In this paper, we introduce the R package **RFpredInterval** (Alakus et al., 2022), which is the novel implementation of 16 methods to build prediction intervals with random forests and boosted forests. The set of methods implemented in the package includes a new method to build prediction intervals with boosted forests and 15 method variations (three splitting rules with the CART paradigm which are LS, L_1 and SPI, and five methods to build prediction intervals which are LM, SPI, Quant, HDR and CHDR) proposed by Roy and Larocque (2020). These 15 methods had been thoroughly investigated before through simulation studies and with real data sets in Roy and Larocque (2020). However, they are not easily available to use. One of the main contributions of our package is the implementation of these competitive methods and the ability for users to compare various prediction interval methods within the same package. The other main contribution of our paper is a new method to build prediction intervals. Contrary to the 15 methods proposed by Roy and Larocque (2020), the newly introduced method was not tested before. That is why in the paper we placed a greater emphasis on investigating the new method through extensive simulation studies and with real data. For performance comparison purposes, we compared the new method to 10 existing methods which include:

- 5 of the 15 implemented method variations of Roy and Larocque (2020); see the Competing methods subsection for details.
- 5 other competing methods from the literature: Quantile Regression Forests (QRF), Generalized Random Forests (GRF), Split Conformal prediction method (SC), Out-of-Bag (OOB) prediction intervals method and $\widehat{PI}_\alpha(x)$ method.

The new proposed method to build **P**rediction **I**ntervals with **B**oosted **F**orests is called **PIBF**. This approach integrates the idea of using the nearest neighbour out-of-bag observations to estimate the conditional prediction error distribution presented in Lu and Hardin (2021) to the one-step boosted forest proposed by Ghosal and Hooker (2021). We will show in this paper that PIBF significantly improves the performance of prediction intervals with random forests when compared with 10 existing methods using a variety of simulated and real benchmark data sets.

The rest of the paper is organized as follows. In the next section, we describe the algorithm implemented in PIBF. We then present the details of the package and provide a practical and reproducible example. We also perform a simulation study to compare the performance of our proposed method to existing competing methods, and we investigate the performance of the proposed method with real data sets. Lastly, we conclude with a discussion of the results.

2 Method and implementation

The proposed method is based on the one-step boosted forest method proposed by Ghosal and Hooker (2021). It consists in fitting two regression random forests: the first is fitted to get point predictions and out-of-bag (OOB) residuals using the given data set, whereas the second is fitted to predict those residuals using the original covariates. As empirical studies demonstrate, the one-step boosted forest provides point predictions with reduced bias compared to the simple random forest. Ghosal and Hooker (2021) use subsampling for their theoretical investigations, even though random forests were originally described with bootstrap samples and obtain notable performance improvements. They

have also investigated the effect of using bootstrapping with the one-step boosted forest on the bias estimations. From the results presented in their appendix, the use of bootstrapping yields the best performance and reduces the bias the most in exchange for an increase in their proposed variance estimator, which is defined under asymptotic normality. In this paper, we use bootstrapping for the one-step boosted forest method following the better performance results on bias reduction. The final prediction for a new observation, x_{new}, is the sum of the predictions from the two random forests

$$\hat{y}^*_{new} = \hat{y}_{new} + \hat{\epsilon}_{new} \tag{2}$$

where $\hat{y}_{new}$ is the point prediction obtained from the first random forest and $\hat{\epsilon}_{new}$ is the bias estimation from the second forest.

Besides bias correction, we use the second random forest as a way to construct a prediction interval by finding the nearest neighbour observations that are close to the one we want to predict. The idea of finding the nearest neighbour observations, a concept very similar to the 'nearest neighbour forest weights' (Hothorn et al., 2004; Lin and Jeon, 2006), was introduced in Moradian et al. (2017) and later used in Moradian et al. (2019), Roy and Larocque (2020), Tabib and Larocque (2020) and Alakuş et al. (2021). For a new observation, the set of in-bag training observations that are in the same terminal nodes as the new observation forms the set of nearest neighbour observations. Roy and Larocque (2020) called this set of observations the Bag of Observations for Prediction (BOP). We can define the BOP for a new observation x_{new} as

$$BOP\left(x_{new}\right) = \bigcup_{b=1}^{B} I_b\left(x_{new}\right) \tag{3}$$

where $I_b\left(x_{new}\right)$ is the set of in-bag training observations, *i.e.*, observations in the bootstrap sample that are in the same terminal node as x_{new} in the b^{th} tree. $I_b\left(.\right)$ consists of the training observations that are in the bootstrap sample of the b^{th} tree.

Instead of forming the set of nearest neighbour observations with the in-bag training observations, we can use the out-of-bag observations which are not in the bootstrap sample, as used in Lu and Hardin (2021). We can define the out-of-bag equivalent of the BOP for a new observation x_{new} (3) as

$$BOP^*\left(x_{new}\right) = \bigcup_{b=1}^{B} O_b\left(x_{new}\right) \tag{4}$$

where $O_b\left(x_{new}\right)$ is the set of out-of-bag observations that are in the same terminal node as x_{new} in the b^{th} tree. $O_b\left(.\right)$ consists of the training observations that are not in the bootstrap sample of the b^{th} tree.

Out-of-bag observations are not used in the tree growing process. Thus, for the trees where the training observations are out-of-bag, they are like the unobserved test observations for those trees. The only difference is that, for a new observation, we use all the trees in the forest whereas for an out-of-bag observation we have only a subset of the forest trees. By using the out-of-bag equivalent of the BOP for a new observation, we can make use of the analogy between the out-of-bag observations and test observations. The out-of-bag neighbours of a new observation represent the new observation better than the in-bag neighbours.

Any desired measure can be obtained by using the constructed BOPs. In this paper, we use the BOP idea to build a prediction interval for a test observation. For a new observation with covariates x_{new}, we firstly form $BOP^*\left(x_{new}\right)$ (4) using the out-of-bag neighbours. Then, as proposed by Lu and Hardin (2021), we estimate the conditional prediction error distribution, $\hat{F}\left(x_{new}\right)$, but now with the bias-corrected out-of-bag residuals of the observations in $BOP^*\left(x_{new}\right)$. Lastly, we build a prediction interval for the new observation as

$$PI\left(x_{new}\right) = \left[\hat{y}^*_{new} + SPI^l_\alpha\left(\hat{F}\left(x_{new}\right)\right), \hat{y}^*_{new} + SPI^u_\alpha\left(\hat{F}\left(x_{new}\right)\right)\right] \tag{5}$$

where $\hat{y}^*_{new}$ is the bias-corrected prediction, $SPI^l_\alpha\left(\hat{F}\left(x_{new}\right)\right)$ and $SPI^u_\alpha\left(\hat{F}\left(x_{new}\right)\right)$ are the lower and upper bounds of the $SPI_\alpha\left(\hat{F}\left(x_{new}\right)\right)$, which is the shortest interval formed by the observations in $BOP^*\left(x_{new}\right)$ that contains at least $\left(1-\alpha\right)100\%$ of the observations. By using the bias-corrected residuals to form prediction error distribution and picking the shortest interval among the qualified intervals, we can expect narrower prediction intervals.

We can summarize the steps of the proposed method as follows:

1. Train the first regression RF with covariates X to predict the response variable Y, and get the OOB predictions $\hat{Y}_{oob}$
2. Compute the OOB residuals as $\hat{\epsilon}_{oob} = Y - \hat{Y}_{oob}$
3. Train the second regression RF with covariates X to predict the OOB residuals $\hat{\epsilon}_{oob}$, and get the

OOB predictions for residuals $\hat{\hat{\epsilon}}_{oob}$

4. Update the OOB predictions as $\hat{Y}^*_{oob} = \hat{Y}_{oob} + \hat{\hat{\epsilon}}_{oob}$
5. Compute the updated OOB residuals after bias-correction as $\hat{\epsilon}^*_{oob} = Y - \hat{Y}^*_{oob}$
6. For a new observation x_{new}, get the point predictions $\hat{y}_{new}$ from the first RF, and get the predicted residuals $\hat{\epsilon}_{new}$ from the second RF, then the final prediction for the new observation is

$$\hat{y}^*_{new} = \hat{y}_{new} + \hat{\epsilon}_{new}$$

where $\hat{\epsilon}_{new}$ is the estimated bias.

7. Form a BOP for x_{new} with the OOB neighbours using the second RF, $BOP^*(x_{new})$ (4) and estimate the conditional prediction error distribution for x_{new} as,

$$\hat{F}(x_{new}) = \left\{\hat{\epsilon}^*_{oob,i} | i \in BOP^*(x_{new})\right\}$$

8. Build a PI for x_{new} as $PI(x_{new}) = \left[\hat{y}^*_{new} + SPI^l_\alpha\left(\hat{F}(x_{new})\right), \hat{y}^*_{new} + SPI^u_\alpha\left(\hat{F}(x_{new})\right)\right]$

Calibration

The principal goal of any prediction interval method is to ensure the desired coverage level. In order to attain the desired coverage level $(1-\alpha)$, we may need a calibration procedure. The goal of the calibration is to find the value of α_w, called the working level in Roy and Larocque (2020), such that the coverage level of the PIs for the training observations is closest to the desired coverage level. Roy and Larocque (2020) presented a calibration procedure that uses the BOPs that are built using only the trees where the training observation x_i is OOB. The idea is to find the value of α_w using the OOB-BOPs. In this paper, we call this procedure OOB calibration.

We also include a cross-validation-based calibration procedure with the proposed method to acquire the desired $(1-\alpha)$ coverage level. In this calibration, we apply k-fold cross-validation to form prediction intervals for the training observations. In each fold, we split the original training data set into training and testing sets. For the training set, we go through the steps 1-5 defined above. Then, for each observation in the testing set, we apply steps 6-8 and build a PI. After completing CV, we compute the coverage level with the constructed PIs and if the coverage is not within the acceptable coverage range, then we apply a grid search to find the α_w such that α_w is the closest to the target α among the set of α_w's. Once we find the α_w, we use this level to build the PI for the new observations.

The RFpredInterval package

In our package, we implement 16 methods that apply random forest training. Ten of these methods have specialized splitting rules in the random forest growing process. These methods are the ones with L_1 and shortest prediction interval (SPI) splitting rules proposed by Roy and Larocque (2020). To implement these methods, we have utilised the custom split feature of the **randomForestSRC** package (Ishwaran and Kogalur, 2021).

The **randomForestSRC** package allows users to define a custom splitting rule for the tree growing process. The user needs to define the customized splitting rule in the `splitCustom.c` file with C-programming. After modifying the `splitCustom.c` file, all C source code files in the package's `src` folder must be recompiled. Finally, the package must be re-installed for the custom split rule to become available.

In our package development process, we froze the version of **randomForestSRC** to the latest one available at the time, which is version 2.11.0, to apply specialized splitting rules. After defining the L_1 and SPI splitting rules, all C files were re-compiled. Finally, all package files including our R files for prediction interval methods were re-built to make the package ready for the user installation.

The **RFpredInterval** package has two main R functions as below:

- `pibf()`: Constructs prediction intervals with the proposed method, PIBF.
- `rfpi()`: Constructs prediction intervals with 15 distinct variations proposed by Roy and Larocque (2020).

Table 1 presents the list of functions and methods implemented in **RFpredInterval**. For `pibf()`, **RFpredInterval** uses the CRAN package **ranger** (Wright et al., 2020) to fit the random forests. For `rfpi()`, **RFpredInterval** uses **randomForestSRC** package. For the least-squares splitting rule, both **randomForestSRC** and **ranger** packages are applicable.

Table 1: List of functions and methods with characteristics

Function	Method		Details
`pibf()`	PIBF		Builds prediction intervals with boosted forests. The **ranger** package is used to fit the random forests. Calibration options are `"cv"` and `"oob"`. Returns constructed PIs and bias-corrected point predictions for the test data.
	Split method	*PI method*	
`rfpi()`	LS	LM SPI Quant HDR CHDR	Splitting rule is the least-squares (LS) from the CART paradigm. `"ls"` is used for the `"split_method"` argument. A vector of characters `c("lm","spi","quant","hdr","chdr")` is used for the `"pi_method"` argument to apply all or a subset of the PI methods. **ranger** or **randomForestSRC** can be used to fit the random forest. Returns a list of constructed PIs for the selected PI methods and point predictions for the test data.
	L_1	LM SPI Quant HDR CHDR	Splitting rule is the L_1 from the CART paradigm (Roy and Larocque, 2020). `"l1"` is used for the `"split_method"` argument. A vector of characters `c("lm","spi","quant","hdr","chdr")` is used for the `"pi_method"` argument to apply all or a subset of the PI methods. Only **randomForestSRC** can be used to fit the random forest since the split rule is implemented with the custom split feature of that package. Returns a list of constructed PIs for the selected PI methods and point predictions for the test data.
	SPI	LM SPI Quant HDR CHDR	Splitting rule is the shortest PI (SPI) from the CART paradigm (Roy and Larocque, 2020). `"spi"` is used for the `"split_method"` argument. A vector of characters `c("lm","spi","quant","hdr","chdr")` is used for the `"pi_method"` argument to apply all or a subset of the PI methods. Only **randomForestSRC** can be used to fit the random forest since the split rule is implemented with the custom split feature of that package. Returns a list of constructed PIs for the selected PI methods and point predictions for the test data.
`piall()`	All methods		Builds prediction intervals with all of the implemented PI methods. The **ranger** package is used to fit the random forests for the PIBF and methods with LS split rule. **randomForestSRC** package is used for the methods with L_1 and SPI split rules. Default values are assigned to the function arguments of `pibf()` and `rfpi()`. Returns an object of class `"piall"` containing a list of constructed PIs with 16 methods, and point predictions obtained with the PIBF method, LS, L_1 and SPI split rules for the test data.
`plot()`			Plots the 16 constructed PIs obtained with `piall()` function for a test observation.
`print()`			Prints the summary output of `pibf()`, `rfpi()` and `piall()` functions.

LM: Classical method, **SPI**: Shortest PI, **Quant**: Quantiles, **HDR**: Highest density region, **CHDR**: Contiguous HDR

In this section, we illustrate the usage of the **RFpredInterval** package with the Ames Housing data set (De Cock, 2011). The data set was introduced as a modern alternative to the well-known Boston Housing data set. The data set contains many explanatory variables on the quality and quantity of physical attributes of houses in Ames, IA sold from 2006 to 2010. Most of the variables give information to a typical home buyer who would like to know about a house (*e.g.* number of bedrooms and bathrooms, square footage, heating type, lot size, etc.).

The **AmesHousing** (Kuhn, 2020) package contains the raw data and processed versions of the Ames Housing data set. The raw data contains 2930 observations and 82 variables, which include 23 nominal, 23 ordinal, 14 discrete, and 20 continuous variables, involved in assessing house values. The processed version of the data set has 2330 observations and 81 variables, including the target variable `Sale_Price` representing the value of houses in US dollars. The usual goal for this data set is to predict the sale price of each house given covariates.

We load the processed version of the Ames Housing data set from the **AmesHousing** package and prepare the data set that we will use for the analyses. The preprocessing steps are presented in the Supplementary Material. This version of the data set contains 22 factors and 59 numeric variables, including 1 response variable `Sale_Price`, for 2929 observations. We split the data set into training and testing samples.

```
set.seed(3456)
n <- nrow(AmesHousing)
trainindex <- sample(1:n, size = round(0.7*n), replace = FALSE)
traindata <- AmesHousing[trainindex, ]
testdata <- AmesHousing[-trainindex, ]
```

We fit a random forest with 1000 trees using the training data and construct 95% prediction intervals for the observations in the testing data with the proposed method. We apply 5-fold cross-validation based calibration and set the acceptable coverage range to $[.945, .955]$. We can pass the list of random forest parameters for **ranger** package.

```
out <- pibf(formula = Sale_Price ~ .,
            traindata = traindata,
            testdata = testdata,
            alpha = 0.05,
```

```
            calibration = "cv",
            numfolds = 5,
            coverage_range = c(0.945, 0.955),
            params_ranger = list(num.trees = 1000),
            oob = TRUE)
```

We can then analyze the constructed PIs and bias-corrected random forest predictions for the testing data, as shown below. The PI output is a list containing lower and upper bounds. For example, we can print the point prediction and prediction interval for the tenth observation in the testing data.

```
out$pred_interval
out$test_pred
c(out$pred_interval$lower[10], out$test_pred[10], out$pred_interval$upper[10])
[1] 133.8629 160.2426 194.5804
```

We can also print the summary output. In the summary output, we can always see the mean PI length over the test data set. If calibration is applied, we can see the working level of α. If the test data set has true response information, as in our example, coverage and prediction errors for the test set are also printed. Moreover, since we have entered `oob = TRUE` in the function arguments, in the summary output we can see the mean PI length and coverage measures along with the prediction errors for the training set. The prediction intervals are built with the out-of-bag (OOB) predictions and prediction errors.

```
print(out)
```

```
>                      alpha_w: 0.050
>                Mean PI length: 73.081
>                      Coverage: 96.8%
>       MAE of test predictions: 12.773
>      RMSE of test predictions: 19.545
>
>      Mean PI length (OOB PIs): 74.823
>            Coverage (OOB PIs): 94.7%
>  MAE of OOB train predictions: 14.179
> RMSE of OOB train predictions: 23.875
```

Next, we construct 95% prediction intervals using the variations proposed by Roy and Larocque (2020). In the following example, the splitting is rule is set to L_1 and we want to apply LM, Quant and SPI methods for building prediction intervals. We apply OOB calibration and set the acceptable coverage range to $[.945, .955]$. We can pass the the list of random forest parameters for **randomForestSRC** package.

```
out2 <- rfpi(formula = Sale_Price ~ .,
             traindata = traindata,
             testdata = testdata,
             alpha = 0.05,
             calibration = TRUE,
             split_rule = "l1",
             pi_method = c("lm", "quant", "spi"),
             params_rfsrc = list(ntree = 1000),
             params_calib = list(range = c(0.945, 0.955)),
             oob = FALSE)
```

We can analyze the constructed PIs for the testing data as below. Each PI output is a list containing lower and upper bounds. For instance, we can print the point prediction and LM prediction interval for the tenth observation in the testing data.

```
out2$lm_interval
out2$quant_interval
out2$spi_interval
c(out2$lm_interval$lower[10], out2$test_pred[10], out2$lm_interval$upper[10])
[1] 129.9474 154.2098 176.8429
```

We print the summary output. In the summary output, we can see the splitting rule selected in the first row. Since the test data set has true responses in our example, we can see the coverage information for the selected PI methods besides the mean PI length and α_w in the printed table. Below the table, we have the mean prediction errors for the test set.

```
print(out2)
```

```
>                   Split rule: L1
> ------------------------------------------------------------------------------
>                                    Mean PI length      Coverage          alpha_w
> Classical method (LM)                  81.641            96.2%            0.140
> Shortest prediction interval (SPI)     81.953            96.1%            0.100
> Quantile method (Quant)                80.893            95.8%            0.120
> ------------------------------------------------------------------------------
>       MAE of test predictions: 14.605
>      RMSE of test predictions: 22.603
```

Although, with the `pibf()` and `rfpi()` functions, we have more flexibility to set the arguments for the methods, we can build prediction intervals with all 16 methods implemented in the package with the `piall()` function. We will build 95% prediction intervals for the test set.

```
out3 <- piall(formula = Sale_Price ~ .,
              traindata = traindata,
              testdata = testdata,
              alpha = 0.05,
              num.trees = 1000)
```

The output is a list of constructed prediction intervals with 16 methods and point predictions obtained with the PIBF method, LS, L_1, and SPI split rules. Hence, the output includes 16 prediction intervals and 4 point predictions which is a list of 20 items in total.

We print the summary output.

```
print(out3)
```

```
> ----------------------------------------
>              Mean PI length    Coverage
> PIBF             72.752          96.6%
> LS-LM            81.800          96.5%
> LS-SPI           82.662          95.4%
> LS-Quant         81.523          95.2%
> LS-HDR           81.733          96.0%
> LS-CHDR          83.025          96.1%
> L1-LM            81.649          96.1%
> L1-SPI           81.805          96.1%
> L1-Quant         80.836          95.3%
> L1-HDR           83.032          96.4%
> L1-CHDR          82.482          96.1%
> SPI-LM           81.578          96.0%
> SPI-SPI          81.960          96.2%
> SPI-Quant        81.036          95.4%
> SPI-HDR          84.404          96.6%
> SPI-CHDR         82.927          96.1%
> ----------------------------------------
>                    MAE            RMSE
> PIBF             12.774         19.428
> LS split         14.412         22.413
> L1 split         14.596         22.569
> SPI split        14.558         22.600
```

Lastly, we plot the constructed prediction intervals with all 16 methods, for the 15th observation in the test set.

```
plot(out3, test_id = 15)
```

Figure 1 presents the prediction intervals and point predictions for the test observation. The methods are ordered in the y-axis based on their resulting PI length. For each method, the red point presents the point prediction and blue lines show the constructed prediction interval(s) for the test observation. If the true response of the test observation is known, it is demonstrated with a dashed vertical line. Note that we may have multiple prediction intervals with the HDR PI method. As we can see from the figure, we may have four different point predictions for the same test observation. The PIBF method and the three splitting rules LS, L_1 and SPI can produce different point predictions. But all PI method variations for the same splitting rule have the same point prediction.

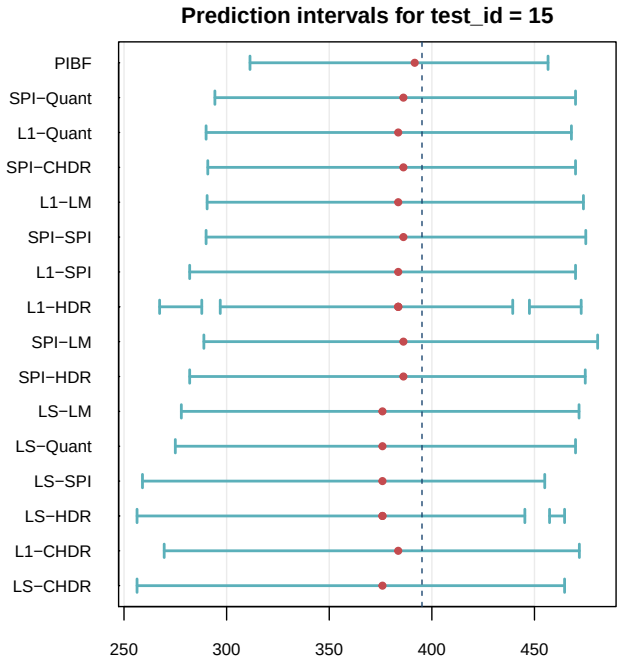


Figure 1: Prediction intervals for the 15th test observation in the test data. The x-axis represents the sale price of houses in thousands. For each method, red dots represent the point prediction and blue lines show the prediction interval(s). The vertical dashed line shows the true response value for the test observation. PIBF: Prediction intervals with boosted forests (the proposed method). The notation for the other 15 methods is *split rule-PI method*. Splitting rules are LS: Least-squares, L1: L_1, SPI: Shortest PI split rule. PI methods are LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR.

3 Simulation study

In this section, we compare the predictive performance of the prediction intervals constructed with our proposed method to the existing methods presented in the Introduction using a variety of simulated and real benchmark data sets.

Simulation design

We apply a simulation study based on seven simulated data sets from the literature. The first three of the data sets are Friedman's benchmark regression problems described in Jerome H. Friedman (1991) and Breiman (1996). We use the CRAN package **mlbench** (Leisch and Dimitriadou, 2021) to generate these data sets.

In Friedman Problem 1, the inputs are 10 independent variables uniformly distributed on the interval $[0, 1]$. The first five covariates are used to generate the response:

$$y = 10 \sin(\pi x_1 x_2) + 20 (x_3 - 0.5)^2 + 10 x_4 + 5 x_5 + \epsilon$$

where ϵ is $N(0, \sigma^2)$ and the default standard deviation of ϵ is 1 which yields a signal-to-noise ratio (SNR) (*i.e.*, the ratio of the standard deviation of signal to the standard deviation of error) of 4.8:1.

In Friedman Problem 2, the response is generated as

$$y = \left(x_1^2 + \left(x_2 x_3 - \frac{1}{x_2 x_4} \right)^2 \right)^{0.5} + \epsilon$$

where the inputs are four independent variables uniformly distributed over the ranges

$$\begin{aligned} 0 \leq & x_1 \leq 100 \\ 40\pi \leq & x_2 \leq 560\pi \\ 0 \leq & x_3 \leq 1 \\ 1 \leq & x_4 \leq 11 \end{aligned}$$

and ϵ is $N\left(0, \sigma^2\right)$. The default value of 125, which yields a SNR of 3:1, is used for the standard deviation of ϵ.

In Friedman Problem 3, the inputs are four independent variables uniformly distributed over the same ranges as Friedman Problem 2. The response is generated as

$$y = \arctan\left(\frac{x_2 x_3 - \frac{1}{x_2 x_4}}{x_1} \right) + \epsilon$$

where ϵ is $N\left(0, \sigma^2\right)$ and the default value of 0.01 for the standard deviation of ϵ is used, which yields a SNR of 3:1.

The fourth data set is the Peak Benchmark Problem which is also from the **mlbench** package. Let $r = 3u$ where u is uniform on $[0, 1]$ and let x be uniformly distributed on the d-dimensional sphere of radius r. The response is $y = 25 \exp\left(-0.5 r^2\right)$. The default value of $d = 20$ dimensions is used.

The fifth one is a modification of Friedman Problem 1, which was used in Hothorn and Zeileis (2021) in their H2c setup. This data set was designed to have heteroscedasticity. The inputs are 10 independent variables uniformly distributed on the interval $[0, 1]$. The first five covariates are used in the mean function and the unscaled mean function is defined as

$$\mu = 10 \sin\left(\pi x_1 x_2\right) + 20 \left(x_3 - 0.5\right)^2 + 10 x_4 + 5 x_5$$

Then, the scaled mean function on the interval $[-1.5, 1.5]$ is

$$\mu^S = \frac{3\left(\mu - \mu_{min}\right)}{\mu_{max} - \mu_{min}} - 1.5$$

where μ_{min} and μ_{max} are the minimum and maximum values of μ over the sample. The last five covariates are used in the standard deviation function and the unscaled standard deviation function is

$$\sigma = 10 \sin\left(\pi x_6 x_7\right) + 20 \left(x_8 - 0.5\right)^2 + 10 x_9 + 5 x_{10}$$

The standard deviation is scaled as

$$\sigma^S = \exp\left(\frac{3\left(\sigma - \sigma_{min}\right)}{\sigma_{max} - \sigma_{min}} - 1.5 \right)$$

where σ_{min} and σ_{max} are the minimum and maximum values of σ over the sample. The response is generated as a normal random variable with mean μ^S and standard deviation σ^S.

The last two data sets, which were used in Roy and Larocque (2020), have a tree-based response variable. The inputs are seven independent variables generated from the standard normal distribution. The response is generated with the seven covariates according to a tree model with a depth of three, with eight terminal nodes:

$$\begin{aligned} y = & u_1 I\left(x_1 < 0, x_2 < 0, x_4 < 0\right) \\ & + u_2 I\left(x_1 < 0, x_2 < 0, x_4 \geq 0\right) \\ & + u_3 I\left(x_1 < 0, x_2 \geq 0, x_5 < 0\right) \\ & + u_4 I\left(x_1 < 0, x_2 \geq 0, x_5 \geq 0\right) \\ & + u_5 I\left(x_1 \geq 0, x_3 < 0, x_6 < 0\right) \\ & + u_6 I\left(x_1 \geq 0, x_3 < 0, x_6 \geq 0\right) \\ & + u_7 I\left(x_1 \geq 0, x_3 \geq 0, x_7 < 0\right) \\ & + u_8 I\left(x_1 \geq 0, x_3 \geq 0, x_7 \geq 0\right) + \epsilon \end{aligned}$$

where the terminal node means are $u = (5, 10, 15, 20, 25, 30, 35, 40)$ and I is the indicator function. The difference in the two data sets is the distribution of the error. In the first one, ϵ is generated from a standard normal distribution and in the other it is from an exponential distribution with mean 1. The signal-to-noise ratio is 11.5:1 for both data sets.

We use training sample sizes of $n_{train} = \{200, 500, 1000, 5000\}$, resulting in 28 scenarios. Each scenario is repeated 500 times. In each run, we generate an independent test set of new observations with $n_{test} = 1000$.

Competing methods

We compare our proposed prediction interval estimator with 10 competing methods which were presented in the Introduction. The first is the $\widehat{PI}_\alpha$ method. We fit the random forest with the **ranger** package and use the **forestError** package to build PIs. The second is the OOB method. The **rfinterval** package is used. The third is the split conformal method. The **conformalInference** package is used. The fourth is the QRF method and the **quantregForest** package is used. The fifth is the GRF method. The function `quantile_forest` in the **grf** package is used.

The last five are variations of Roy and Larocque (2020). To compare the performance of the method variations, a comprehensive simulation study and real data analyses were performed in Roy and Larocque (2020). One of the biggest conclusions from these comparison studies was that, among the three alternative splitting criteria within the CART paradigm, the impact of the choice of the splitting rule on the performance of the prediction intervals was moderate whereas the selection of the PI method had a much greater impact on the performance. Hence, in this simulation study, we set up the splitting rule to the least-squares (LS) and only compare the five PI methods: LM, Quant, SPI, HDR, and CHDR. For those methods, the `rfpi()` function of the **RFpredInterval** package is used. We fit the random forest with the **ranger** package.

Parameter settings

For the simulations, we use the following parameters. For all methods, we set the number of trees to 2000. Letting p be the number of covariates, then the number of covariates to randomly split at each node, *mtry*, is set to $\max\{\lfloor p/3 \rfloor, 1\}$ (except for the GRF method). For the GRF method, following Athey et al. (2019), *mtry* is set to $\min\{\lceil \sqrt{p} + 20 \rceil, p\}$. Also, we use the honest splitting regime with the default fraction of 0.5 for the GRF method. The minimum node size parameter for all forests is set to 5. The desired coverage is set to 95% for all methods. For the proposed method, we perform the cross-validation-based calibration as the primary calibration procedure, but we also investigate the OOB calibration. For the method variations in Roy and Larocque (2020), we perform the OOB calibration procedure as they proposed. For both calibration procedures, the acceptable range of coverage is set to $[.945, .955]$. Calibration is not performed for the competing methods since no option for calibration is offered in their CRAN packages.

Performance with the simulated data sets

We can evaluate the performance of the competing methods with two measures: the mean coverage and the mean prediction interval length. Table 2 presents the average coverage rate of each method on the test set over 500 replications for a given simulated data set and sample size, with average mean prediction interval lengths shown in parentheses. The principal goal of any prediction interval method is to ensure the desired coverage level. In this simulation study, the desired coverage level is set to 95% for all methods. The left plot in Figure 2 shows the mean coverages over the 28 scenarios for all methods. Overall, all of the methods, except the QRF and GRF methods which tend to be conservative, provide a mean coverage close to the desired level. QRF and GRF methods have an average mean coverage of 0.975 and 0.974 over all scenarios, respectively. Although the $\widehat{PI}_\alpha$ method has an average of the mean coverages of 0.957, close to the desired level, its variability is large. Over 308 (11 methods $\times$ 28 scenarios) average coverage values, there is only one case where the mean coverage is below 0.94. It corresponds to the $\widehat{PI}_\alpha$ method in Friedman Problem 2 with $n_{train} = 5000$.

Once the prediction intervals provide the desired coverage level, the next goal of any PI method is to provide the shortest PI length. Prior to carrying out a detailed comparison of interval lengths, we can globally compare the interval lengths over all scenarios with the percentage increase in mean PI length of a method with respect to the best method for a given run. For a given run, define ml_i as the mean PI length of method i and ml^* as the shortest mean PI length over the 10 competing methods. The percentage increase in PI length for method i is computed as $100 \times (ml_i - ml^*)/ml^*$. Smaller values for this measure indicate better performances. The right plot in Figure 2 presents the relative

lengths of the methods across 14,000 runs (28 scenarios $\times$ 500 replications). The prediction intervals with the GRF method are the widest, followed by the QRF method. However, as we saw in the left plot in Figure 2, GRF and QRF produce conservative prediction intervals, so their PI lengths cannot be fairly compared to the other methods with a coverage closer to 0.95. Based on the global results, the proposed method, PIBF, performs the best. Following PIBF, $\widehat{PI}_\alpha$, OOB, LM, SPI, HDR and CHDR perform similarly well, with $\widehat{PI}_\alpha$ being slightly better. Among the variations of Roy and Larocque (2020), the PIs with quantiles produce longer prediction intervals than the other four variations.

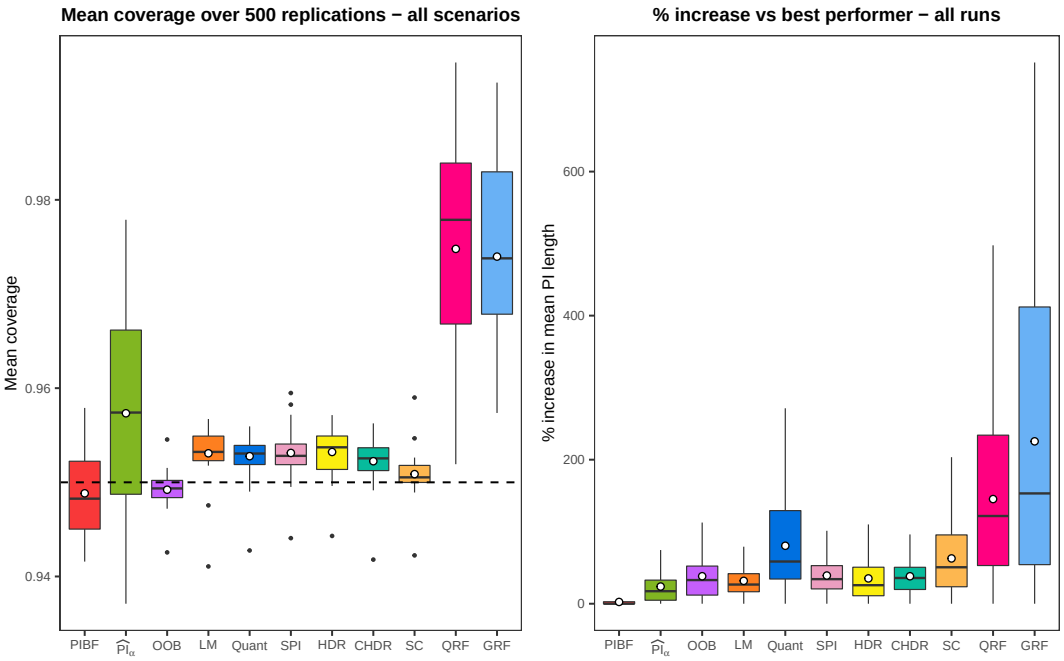


Figure 2: *(Left)* Boxplots for the mean coverage over all scenarios. All methods, except the QRF and GRF methods, are able to provide a mean coverage close to the desired coverage level of 0.95. Each white circle is the average of the mean coverages over 28 scenarios. *(Right)* Boxplots for the percentage increase in mean PI length of each method compared to the shortest PI length for a given run across 14,000 runs. The smallest is the percentage increase, the better is the method. Each white circle is the average of the relative lengths over 14,000 runs. Since the outlier values are distorting the scales, they are removed from the graph. PIBF: Prediction intervals with boosted forests (the proposed method), $\widehat{PI}_\alpha$: Conditional α-level prediction interval, OOB: Out-of-Bag approach, LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR, SC: Split conformal, QRF: Quantile regression forest, GRF: Generalized random forest.

Now, we investigate the performance of the methods separately for each scenario. See figures S1 to S7 in the Supplementary Material for the mean PI length results of each method for each simulated data set. Each figure has four facets corresponding to the four levels of the training sample size. For all methods and data sets, the mean PI lengths and their variability decrease as the sample size increases (except the GRF method for the tree-based data sets). We see that for Friedman Problem 1, from Figure S1, for all sample sizes, PIBF consistently outperforms the 10 competing methods in terms of mean PI length while ensuring the desired coverage level (see Table 2 for the mean coverage results). QRF and GRF provide the widest prediction intervals for all sample sizes. However, as presented in Table 2, these methods heavily over-cover and are therefore not comparable with the other methods. While the $\widehat{PI}_\alpha$ method also slightly over-covers, it has shorter PIs than other methods.

For Friedman Problem 2 (see Figure S2 in the Supplementary Material), we see that the proposed method has the shortest mean PI length for the smallest sample size, and as the sample size increases $\widehat{PI}_\alpha$ provides shorter PIs. However, we should also take into account the mean coverages presented in Table 2. The proposed method has smaller coverage levels for $n_{train} = 200$ compared to $\widehat{PI}_\alpha$, but as the sample size increases the coverage levels decrease for the $\widehat{PI}_\alpha$ method (up to 0.937 for $n_{train} = 5000$) whereas PIBF keeps it around 0.945. For $n_{train} = 5000$, the OOB method builds shorter PIs while ensuring the desired coverage level. Again, QRF and GRF have the widest PIs for all sample sizes due to their conservative PIs.

The performance of PIBF and $\widehat{PI}_\alpha$ is very similar for Friedman Problem 3 (see Figure S3 in the Supplementary Material). Both methods provide the shortest PIs with similar coverage levels and

Table 2: Results of the simulation study. Average coverage rates of each method in each simulation, with average mean PI lengths shown in parentheses. The desired coverage level is 0.95. Shortest average mean PI lengths are emphasized with bold text. PIBF: Prediction intervals with boosted forests (the proposed method), $\widehat{PI}_\alpha$: Conditional α-level prediction interval, OOB: Out-of-Bag approach, LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR, SC: Split conformal, QRF: Quantile regression forest, GRF: Generalized random forest.

n_{train}	Data	PIBF	$\widehat{PI}_\alpha$	OOB	LM	Quant	SPI	HDR	CHDR	SC	QRF	GRF
200	Friedman 1	0.952 (**7.69**)	0.959 (9.7)	0.949 (10.3)	0.955 (11.2)	0.956 (12.5)	0.956 (12.1)	0.955 (12.2)	0.956 (11.5)	0.952 (12)	0.978 (15.3)	0.968 (17.7)
	Friedman 2	0.942 (**635**)	0.951 (679)	0.947 (762)	0.956 (772)	0.955 (990)	0.956 (898)	0.955 (793)	0.954 (800)	0.951 (912)	0.969 (1103)	0.972 (1241)
	Friedman 3	0.944 (**0.57**)	0.948 (0.62)	0.949 (0.75)	0.956 (0.69)	0.953 (0.83)	0.952 (0.74)	0.951 (0.88)	0.954 (0.76)	0.953 (0.93)	0.961 (0.89)	0.968 (0.88)
	Peak	0.958 (**9.06**)	0.956 (12.4)	0.949 (13.3)	0.955 (12)	0.956 (17.9)	0.957 (10.5)	0.955 (15.4)	0.955 (11.1)	0.951 (15.5)	0.972 (20.6)	0.978 (20.2)
	H2c	0.954 (6.5)	0.954 (6.3)	0.955 (6.26)	0.955 (**5.86**)	0.956 (6.42)	0.957 (6.47)	0.956 (6.61)	0.955 (6.45)	0.959 (6.68)	0.952 (6.09)	0.957 (6.35)
	Tree-N	0.949 (9.12)	0.967 (12.3)	0.947 (14.2)	0.956 (11.9)	0.956 (22.3)	0.958 (15.4)	0.955 (**9.08**)	0.955 (11.9)	0.951 (18.9)	0.979 (29.6)	0.960 (36)
	Tree-exp	0.949 (**9.31**)	0.967 (12.4)	0.947 (14.2)	0.955 (11.9)	0.956 (22.1)	0.959 (15.1)	0.956 (10.4)	0.956 (12.5)	0.950 (18.9)	0.979 (29.5)	0.960 (35.8)
500	Friedman 1	0.952 (**6.32**)	0.962 (8.25)	0.948 (8.81)	0.953 (9.37)	0.952 (10.1)	0.953 (9.87)	0.952 (9.81)	0.953 (9.47)	0.951 (10.1)	0.986 (14)	0.978 (16.9)
	Friedman 2	0.945 (586)	0.946 (**585**)	0.948 (650)	0.952 (676)	0.952 (820)	0.952 (744)	0.951 (679)	0.951 (690)	0.952 (751)	0.975 (998)	0.979 (1120)
	Friedman 3	0.943 (**0.5**)	0.946 (0.53)	0.948 (0.62)	0.952 (0.61)	0.951 (0.71)	0.951 (0.65)	0.951 (0.7)	0.951 (0.67)	0.952 (0.75)	0.965 (0.79)	0.970 (0.77)
	Peak	0.957 (**6.44**)	0.967 (9.97)	0.951 (11)	0.953 (9.75)	0.954 (15)	0.954 (8.22)	0.956 (11.6)	0.953 (8.93)	0.952 (12.8)	0.980 (18.7)	0.984 (17.8)
	H2c	0.953 (5.81)	0.956 (5.88)	0.951 (5.89)	0.952 (**5.43**)	0.953 (5.83)	0.953 (5.81)	0.954 (5.92)	0.952 (5.84)	0.955 (6.17)	0.954 (5.79)	0.959 (5.94)
	Tree-N	0.948 (**6.16**)	0.969 (8.17)	0.949 (9.85)	0.952 (8.23)	0.953 (15.5)	0.953 (10)	0.954 (7.31)	0.952 (9.6)	0.952 (13.3)	0.984 (25.8)	0.969 (35.7)
	Tree-exp	0.947 (**6.31**)	0.966 (8.3)	0.949 (9.93)	0.952 (8.19)	0.953 (15.4)	0.954 (9.67)	0.954 (7.93)	0.953 (9.96)	0.953 (13.4)	0.984 (25.8)	0.970 (35.5)
1000	Friedman 1	0.953 (**5.67**)	0.964 (7.42)	0.949 (7.95)	0.954 (8.37)	0.954 (8.89)	0.954 (8.75)	0.954 (8.65)	0.953 (8.4)	0.950 (8.85)	0.990 (12.9)	0.985 (15.9)
	Friedman 2	0.945 (565)	0.942 (**546**)	0.950 (594)	0.953 (635)	0.953 (735)	0.953 (682)	0.952 (637)	0.953 (645)	0.951 (658)	0.977 (910)	0.983 (1022)
	Friedman 3	0.944 (**0.47**)	0.945 (0.48)	0.948 (0.55)	0.954 (0.58)	0.952 (0.65)	0.952 (0.61)	0.951 (0.63)	0.952 (0.63)	0.949 (0.63)	0.967 (0.73)	0.971 (0.72)
	Peak	0.957 (**5**)	0.972 (8.44)	0.950 (9.54)	0.952 (8.51)	0.953 (13.2)	0.953 (7.21)	0.956 (9.35)	0.952 (7.82)	0.951 (11)	0.983 (17.5)	0.986 (16.7)
	H2c	0.952 (5.48)	0.955 (5.64)	0.949 (5.69)	0.948 (**5.17**)	0.949 (5.49)	0.950 (5.51)	0.950 (5.49)	0.949 (5.49)	0.949 (5.78)	0.954 (5.61)	0.958 (5.71)
	Tree-N	0.946 (**5.11**)	0.965 (6.2)	0.950 (7.52)	0.954 (6.67)	0.953 (12.1)	0.954 (7.91)	0.954 (6.21)	0.953 (8.33)	0.950 (9.94)	0.985 (21.6)	0.976 (35.1)
	Tree-exp	0.946 (**5.16**)	0.964 (6.39)	0.949 (7.64)	0.954 (6.62)	0.953 (11.9)	0.954 (7.45)	0.955 (6.5)	0.953 (8.45)	0.950 (10)	0.985 (21.6)	0.975 (34.9)
5000	Friedman 1	0.950 (**4.71**)	0.967 (6.04)	0.950 (6.47)	0.953 (6.67)	0.953 (6.97)	0.953 (6.91)	0.954 (6.78)	0.953 (6.67)	0.950 (7.03)	0.995 (10.8)	0.992 (13.2)
	Friedman 2	0.945 (543)	0.937 (**507**)	0.950 (529)	0.952 (574)	0.951 (610)	0.951 (592)	0.951 (570)	0.951 (573)	0.950 (549)	0.975 (717)	0.982 (781)
	Friedman 3	0.945 (0.44)	0.941 (**0.43**)	0.950 (0.46)	0.953 (0.53)	0.952 (0.56)	0.952 (0.55)	0.951 (0.54)	0.952 (0.56)	0.950 (0.49)	0.966 (0.62)	0.971 (0.62)
	Peak	0.955 (**2.84**)	0.978 (5.92)	0.952 (7.11)	0.952 (6.54)	0.954 (10.1)	0.952 (5.68)	0.957 (6.46)	0.951 (6.04)	0.950 (8)	0.988 (14.9)	0.990 (15.2)
	H2c	0.949 (5)	0.953 (5.31)	0.943 (5.39)	0.941 (**4.82**)	0.943 (5.01)	0.944 (5.06)	0.944 (5.03)	0.942 (4.98)	0.942 (5.41)	0.953 (5.33)	0.958 (5.31)
	Tree-N	0.945 (**4.33**)	0.949 (4.37)	0.951 (4.77)	0.957 (5.25)	0.951 (7.03)	0.950 (5.4)	0.953 (4.78)	0.951 (5.89)	0.950 (5.55)	0.979 (10.5)	0.986 (32.1)
	Tree-exp	0.945 (**4.08**)	0.959 (4.39)	0.950 (4.91)	0.955 (5.12)	0.951 (6.87)	0.952 (4.78)	0.954 (4.36)	0.949 (5.49)	0.950 (5.73)	0.978 (10.5)	0.986 (31.8)

mean PI lengths. Results for the Peak Benchmark Problem presented in Figure S4 are very similar to those of Friedman Problem 1. For all sample sizes, PIBF consistently outperforms the 10 competing methods in terms of mean PI length. But this time, SPI method with LS splitting rule comes in second place.

For the H2c setup (see Figure S5 in the Supplementary Material), which is the modification of Friedman Problem 1, we can see that all methods are comparable since QRF and GRF do not over-cover. In this setting, all methods also perform fairly well with respect to PI length. Overall, the LM prediction interval method with the LS splitting rule provides slightly shorter PIs.

For the tree-based data sets (see figures S6 and S7 in the Supplementary Material), overall, it seems that the distribution of the error does not have a significant effect on the results. Again, QRF and GRF have conservative PIs. Unlike the other data sets, we see here that the mean PI lengths of the GRF method decrease very slowly as the sample size increases. For $n_{train} = 5000$, all methods (except QRF and GRF) perform similarly. For the smallest sample size, HDR PI building methods and the proposed method perform slightly better than other methods. As the sample size increases, PIBF produces the shortest prediction intervals.

Effect of calibration on the performance of prediction intervals

In this section, we investigate the effect of the proposed calibration on performance of the prediction intervals. We apply the same simulation study using the seven simulated data sets. We compare the results of the proposed method without calibration, with OOB calibration and calibration with cross-validation. The desired coverage level is set to 95%. In Figure 3, the left plot presents the mean coverages over 28 scenarios for the three variations, and the right plot shows the percentage increase in mean PI length of each of the three calibration variants across 14,000 runs (28 scenarios $\times$ 500 replications). Although we obtain the shortest prediction intervals without calibration, the variability of the mean coverage level is larger and sometimes the coverage falls below 0.94. Looking at the left plot, we can say that the variability of the mean coverage level decreases with both calibration procedures. However, applying OOB calibration provides conservative PIs. The median of the mean coverage level is more than 0.96 and the PIs with the OOB calibration are the widest. Applying calibration with CV produces slightly longer PIs than those with no calibration, but these PIs have coverage levels closer to the desired level.

In the package, both calibration procedures are implemented for the PIBF method. Simulation study results show that, compared to the OOB calibration, calibration with CV produces shorter PIs while maintaining the desired coverage level. Therefore, the default calibration procedure is set to CV in the `pibf()` function. In terms of computational time (see tables 4 and 5), calibration with k-fold CV is slower than OOB calibration since it needs to fit two additional random forests for each fold.

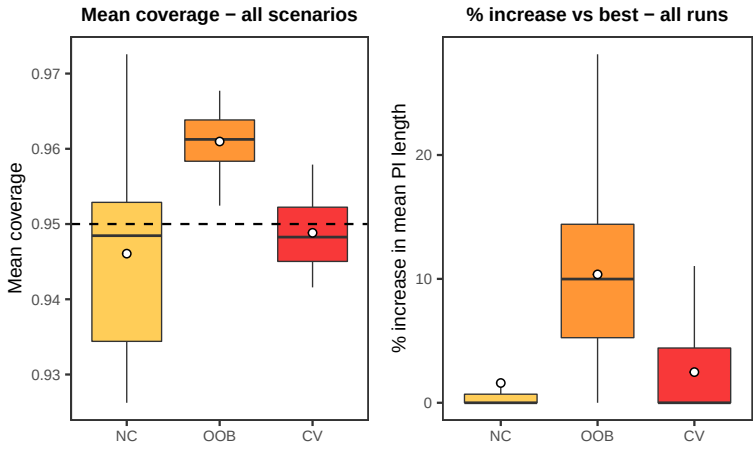


Figure 3: Global performance results of the proposed method for the simulated data sets with different calibration procedures. NC: No calibration, OOB: OOB calibration, CV: Calibration with cross-validation. *(Left)* Boxplots for the mean coverage over 500 replications across all scenarios. *(Right)* Boxplots for the percentage increase in mean PI length of each calibration procedure compared to the shortest PI length for a given run across 14,000 runs. Smaller values are better. Since outlier values are distorting the scales, they are removed.

Performance with real data sets

To further explore the performance of the prediction intervals built with the proposed method, we use 11 real data sets. Since two of the data sets have two response variables, we consider that we are analyzing 13 real data sets. Boston housing and Ames housing data sets are from the R packages **mlbench** and **AmesHousing**, respectively. The other data sets are obtained through the UCI Machine Learning Repository (Dua and Graff, 2017).

For each data set, we apply 100 times 10-fold cross-validation for each method. Hence, for each fold, the training and testing sets correspond to 90% and 10% of the whole data set, respectively. The desired coverage level is set to 95% for all methods. Table 3 presents the results of the real data analyses (n is the number of observations and p is the number of predictors) with the average coverage rate of each method over 100 repetitions, and mean prediction interval lengths averaged over 100 repetitions shown in parentheses. Figure 4 illustrates the global results of the analyses across datasets. The left plot in Figure 4 shows the mean coverages over the 13 real data sets for all methods. Similar to what we have seen with the simulated data sets, the QRF and GRF methods produce conservative prediction intervals, whereas the other methods provide a mean coverage close to the desired level. Again, the $\widehat{PI}_\alpha$ method maintains the target level on average with 0.959, but its variability is the highest among all methods. Across all data sets, there are three cases where the mean coverage is below 0.94: the proposed method for Concrete slump, and the $\widehat{PI}_\alpha$ method for Auto MPG and Computer hardware.

The right plot in Figure 4 presents the relative lengths of methods across 13 real data sets. For each method, there are 13 points in the boxplot, and each point corresponds to the percentage increase in mean PI length compared to the best method for a single real data set. Again, the prediction intervals with GRF and QRF are the widest among eleven methods. Among the other nine methods, the proposed method performs the best, followed by $\widehat{PI}_\alpha$.

For each real data set, we analyze the performance of each method through the mean PI lengths presented in figures S8 to S10 in the Supplementary Material. For the Abalone data set, the HDR method produces the shortest prediction intervals, followed by the SPI and Quant methods. While the QRF method over-covers, its PIs are no wider than those of most of the other methods. The proposed method, PIBF, is distinctly the best prediction interval method yielding the shortest PI lengths for the Air quality data set with absolute and relative humidity response variables, Airfoil selfnoise, Ames housing, Boston housing, Concrete compression, Energy efficiency data set with cooling and heating load response variables, and Servo data sets. In the Auto MPG data set, $\widehat{PI}_\alpha$ has the shortest mean

PI length but with a mean coverage of 0.929. Among the other methods, PIBF, OOB, LM, Quant and SPI methods show similarly good performances while maintaining the desired coverage level. For the Computer hardware data set, PIBF, $\widehat{PI}_\alpha$, and SPI methods perform better than the other methods. They have similar mean PI lengths. For the Concrete slump data set, the proposed method has the shortest mean PI length, but with a slightly smaller coverage of 0.939. This data set is the only one of the simulated and real data sets where the proposed method has a mean coverage below 0.94. After PIBF, $\widehat{PI}_\alpha$ and LM show a good performance with a mean coverage close to the target level. Overall, we can conclude that the proposed method shows better performance than the 10 competing methods for almost all of the real data sets.

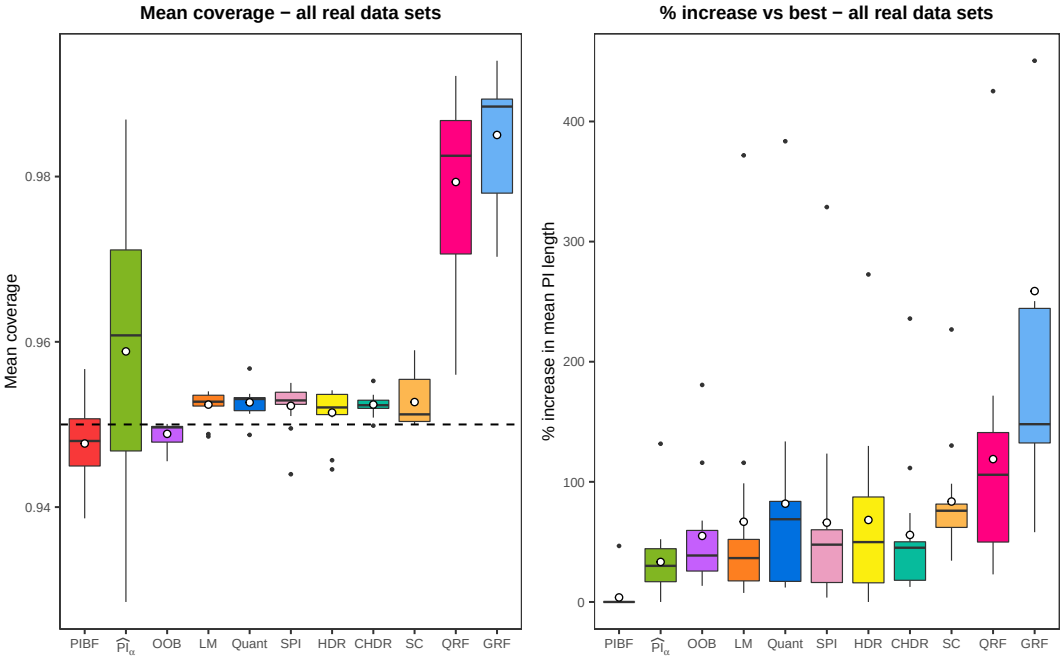


Figure 4: *(Left)* Boxplots for the mean coverage over all real data sets. All methods except the QRF and GRF methods are able to provide a mean coverage close to the desired coverage level of 0.95. Each white circle is the average of the mean coverages over 13 real data sets. *(Right)* Boxplots for the percentage increase in mean PI length of each method compared to the shortest PI length for a given real data set across 13 data sets. The smallest the percentage increase, the better the method. Each white circle is the average of the relative lengths over 13 real data sets. One of the outlier values for GRF with the percentage increase of 1244% is removed from the graph since it is distorting the scales. PIBF: Prediction intervals with boosted forests (the proposed method), $\widehat{PI}_\alpha$: Conditional α-level prediction interval, OOB: Out-of-Bag approach, LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR, SC: Split conformal, QRF: Quantile regression forest, GRF: Generalized random forest.

Comparison of the computational times

All simulations and real data analyses were conducted in R version 3.6.0 on a Linux machine with Intel(R) Xeon(R) E5-2667 v3 @ 3.20GHz with 396 GB of memory. The average computational time of each method for the simulated and real data sets are presented in tables 4 and 5. For the proposed method (PIBF), the computational times for both calibration methods, cross-validation and OOB, are presented in the tables. We can see from the tables that, for most of the data sets, calibration with cross-validation has longer running times than OOB calibration, which is expected since with the k-fold cross-validation, we fit $2k$ more random forests than applying OOB calibration.

For the variations of Roy and Larocque (2020), since we can build prediction intervals with the five PI methods by only fitting a single random forest with a selected splitting rule, we present the total computational time for building the five variations under RFPI. To be clear, for a given splitting rule, the `rfpi()` function fits a random forest and then the set of PI methods requested by the user are applied to the output of the random forest. In our simulations, we choose to return all five PI methods for the selected splitting rule, *i.e.* when we measure the running time of the `rfpi()` function, we get the total running time of building five prediction intervals. Therefore, we should interpret the

Table 3: Results of the real data analysis. Average coverage rates of each method for each real data set, with average mean PI lengths shown in parentheses. The desired coverage level is 0.95. Shortest average mean PI lengths are shown in bold. PIBF: Prediction intervals with boosted forests (the proposed method), $\widehat{PI}_\alpha$: Conditional α-level prediction interval, OOB: Out-of-Bag approach, LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR, SC: Split conformal, QRF: Quantile regression forest, GRF: Generalized random forest.

Data	n	p	PIBF	$\widehat{PI}_\alpha$	OOB	LM	Quant	SPI	HDR	CHDR	SC	QRF	GRF
Abalone	4177	8	0.947 (8.18)	0.946 (8.05)	0.950 (8.9)	0.949 (8.09)	0.953 (6.81)	0.951 (6.73)	0.946 (**5.58**)	0.951 (8.12)	0.950 (9.04)	0.971 (8.03)	0.978 (8.82)
Air quality (AH)	6941	13	0.948 (**0.22**)	0.966 (0.29)	0.950 (0.34)	0.954 (0.29)	0.954 (0.33)	0.954 (0.31)	0.954 (0.34)	0.954 (0.29)	0.950 (0.4)	0.992 (0.55)	0.994 (0.78)
Air quality (RH)			0.949 (**12.9**)	0.971 (18.7)	0.950 (21.6)	0.953 (19.6)	0.953 (22.5)	0.954 (20.6)	0.954 (21.1)	0.953 (19.3)	0.950 (25.5)	0.989 (35)	0.989 (44.3)
Airfoil selfnoise	1503	5	0.946 (**8.47**)	0.972 (12.9)	0.950 (13.3)	0.952 (18.3)	0.952 (19.8)	0.953 (18.9)	0.952 (19.5)	0.953 (17.9)	0.951 (14.7)	0.987 (20.4)	0.988 (18.1)
Ames housing	2929	80	0.955 (**69.3**)	0.961 (81.6)	0.950 (90.3)	0.954 (79.3)	0.953 (80.5)	0.953 (80.5)	0.952 (80.3)	0.952 (80.4)	0.951 (96.4)	0.987 (118)	0.993 (172)
Auto MPG	392	7	0.945 (10)	0.929 (**9.76**)	0.947 (11.1)	0.953 (10.5)	0.953 (10.9)	0.953 (10.5)	0.954 (13.2)	0.952 (11.5)	0.955 (13.1)	0.967 (12)	0.977 (16.1)
Boston housing	506	13	0.942 (**10.5**)	0.948 (11.2)	0.949 (12.3)	0.954 (11.7)	0.953 (11.8)	0.953 (11.4)	0.952 (12.2)	0.952 (12)	0.951 (15)	0.982 (15.7)	0.990 (25.9)
Computer hardware	209	6	0.942 (**139**)	0.939 (140)	0.946 (171)	0.952 (163)	0.951 (163)	0.950 (144)	0.951 (157)	0.950 (156)	0.957 (248)	0.965 (182)	0.985 (365)
Concrete compression	1030	8	0.948 (**14.8**)	0.957 (18)	0.950 (19.6)	0.954 (20.2)	0.953 (27.2)	0.954 (22.8)	0.953 (27.7)	0.953 (21.5)	0.950 (26)	0.982 (34.5)	0.989 (46.6)
Concrete slump	103	7	0.939 (**12.2**)	0.947 (14.2)	0.948 (15.3)	0.949 (14.9)	0.949 (20.6)	0.944 (19.4)	0.945 (16.3)	0.951 (15.4)	0.958 (22.1)	0.956 (22.3)	0.970 (28.3)
Energy efficiency (CL)	768	8	0.953 (**3.71**)	0.977 (5)	0.950 (8.01)	0.952 (7.37)	0.952 (7.9)	0.952 (7.04)	0.951 (8.03)	0.953 (6.46)	0.951 (8.54)	0.985 (8.18)	0.986 (20.4)
Energy efficiency (HL)			0.951 (**1.48**)	0.987 (3.43)	0.950 (4.16)	0.953 (6.99)	0.952 (7.17)	0.954 (6.35)	0.951 (5.52)	0.952 (4.98)	0.952 (4.84)	0.988 (7.79)	0.989 (19.9)
Servo	167	4	0.957 (**18.4**)	0.966 (24.2)	0.948 (25.5)	0.953 (26.6)	0.957 (32.9)	0.955 (27.2)	0.954 (28.2)	0.955 (26.8)	0.959 (30.4)	0.983 (37.9)	0.977 (42.9)

values for RFPI with care while comparing the computational times of the methods. Although not all PI methods have similar computational complexities, we can say that even the average time of building prediction intervals with one of these variations, assuming they have similar running times, is reasonable. Since, for the HDR-based PI methods, an optimal bandwidth has to be chosen, which is a time-consuming process, among the five PI methods, the slowest ones are the HDR and CHDR. From the remaining variations, the classical method, LM, is the fastest, followed by the Quant and SPI methods.

For both the simulations and real data analyses, the OOB and GRF methods have the smallest running times. For most of the methods, the increase in the sample size has a mild effect on the ratio of increase in running times. However, for the split conformal method with the simulated data sets, running times increase more than the proportional increase in sample sizes. Similarly, we can see that the QRF method is also affected from the training sample size as it rises to 5000.

4 Concluding remarks

In this paper, we have introduced an R package named **RFpredInterval**. This package implements 16 methods to build prediction intervals with random forests: a new method to build **P**rediction **I**ntervals with **B**oosted **F**orests (PIBF) and 15 different variations to produce prediction intervals with random forests proposed by Roy and Larocque (2020). PIBF provides bias-corrected point predictions obtained with the one-step boosted forest and prediction intervals by using the nearest neighbour out-of-bag observations to estimate the conditional prediction error distribution.

We performed an extensive simulation study with a variety of simulated data sets and applied real data analyses to investigate the performance of the proposed method. The performance was evaluated based on the coverage level and length of the prediction intervals. We compared the performance of the proposed method to 10 existing methods for building prediction intervals with random forests. The proposed method was able to maintain the desired coverage level with both the simulated and real data sets. In terms of the PI lengths, globally, the proposed method provided the shortest prediction intervals among all methods. The conclusions drawn from the analysis of real data sets were very similar to those with the simulated data sets. This provides evidence for the reliability of the proposed method. All results obtained indicate that the proposed method can be used with confidence for a variety of regression problems.

Note that the coverage rate of prediction intervals for new observations can have several interpretations. An interesting discussion about this issue is given in Mayr et al. (2012). In that paper, the authors presented two interpretation of coverage: sample coverage and conditional coverage. Sample coverage means that if we draw a new sample from the same population as the training sample and build PIs with a desired coverage level of $(1-\alpha)$, then the global coverage rate over this sample will be $(1-\alpha)$. The conditional coverage means that if we sample many new observations always having the same set of covariates and build PIs for them with a desired coverage level of $(1-\alpha)$, then about $(1-\alpha)\,100\%$ of these prediction intervals will contain the true value of the response. To hold a desired level of conditional coverage, the predictive method needs to provide the desired coverage level over the entire covariate space. On the other hand, sample coverage needs only maintain the desired coverage level over the new sample, on average. Therefore, if the conditional coverage holds, then the sample coverage also holds. In practice, predictive models are mostly evaluated with their

Table 4: Average computational time (in seconds) of each method over 500 replications for each simulated data set. The values represent the average time to build prediction intervals for a test set with 1000 observations. PIBF-CV: The proposed method with the cross-validation calibration, PIBF-OOB: The proposed method with the OOB calibration, RFPI: The average total running time of the five PI methods, *i.e.* LM + Quant + SPI + HDR + CHDR.

n_{train}	Data	PIBF-CV	PIBF-OOB	$\widehat{PI}_\alpha$	OOB	RFPI	SC	QRF	GRF
200	Friedman 1	21.36	13.44	9.62	0.25	87.79	0.61	3.05	0.27
	Friedman 2	21.83	13.96	9.62	0.23	59.50	0.44	2.61	0.26
	Friedman 3	28.84	16.03	11.46	0.20	75.47	0.41	2.50	0.22
	Peak	18.76	10.95	11.91	0.28	73.15	0.89	4.05	0.43
	H2c	12.92	9.83	7.99	0.27	36.94	0.79	4.05	0.30
	Tree-N	22.75	21.17	12.21	0.23	100.02	0.50	2.81	0.26
	Tree-exp	16.28	15.62	15.65	0.23	87.49	0.53	2.85	0.25
500	Friedman 1	38.37	14.63	8.50	0.44	96.18	2.45	9.29	0.66
	Friedman 2	27.07	13.64	7.64	0.43	84.19	1.80	6.79	0.47
	Friedman 3	18.85	17.49	7.98	0.47	70.78	1.86	4.96	0.45
	Peak	29.16	18.14	9.72	0.70	212.23	2.29	7.49	1.26
	H2c	14.84	11.29	11.20	0.44	58.60	1.74	4.95	0.78
	Tree-N	17.80	20.31	7.70	0.51	183.92	2.21	4.28	0.51
	Tree-exp	18.05	19.05	7.74	0.51	170.24	1.72	4.45	0.49
1000	Friedman 1	32.92	18.07	12.11	0.64	181.30	3.35	9.54	1.50
	Friedman 2	23.07	15.94	10.25	0.45	135.64	2.14	6.26	0.90
	Friedman 3	23.24	16.07	7.37	0.44	96.76	2.27	6.40	0.69
	Peak	51.93	33.12	7.95	0.83	374.89	5.35	19.71	1.65
	H2c	26.81	12.56	8.07	0.59	80.38	3.68	13.11	0.87
	Tree-N	22.84	16.70	7.11	0.47	288.35	2.72	9.53	0.78
	Tree-exp	21.06	17.26	7.03	0.48	272.52	2.62	10.38	0.69
5000	Friedman 1	155.81	139.88	28.70	3.72	928.84	40.79	134.97	4.84
	Friedman 2	155.62	101.58	35.25	2.30	430.80	33.43	60.99	2.65
	Friedman 3	106.70	91.72	34.41	2.36	330.81	28.47	73.18	2.72
	Peak	287.67	245.17	22.76	6.84	1914.56	57.96	123.19	11.80
	H2c	283.53	107.57	24.12	4.31	271.18	42.69	79.19	5.01
	Tree-N	106.17	71.78	22.97	3.19	753.32	26.88	74.81	3.86
	Tree-exp	95.85	68.90	21.76	3.32	779.42	26.09	64.02	3.85

Table 5: Average computational time (in seconds) of each method over 100 times 10-fold cross-validation for each real data set. PIBF-CV: The proposed method with the cross-validation calibration, PIBF-OOB: The proposed method with the OOB calibration, RFPI: The average total running time of the five PI methods, *i.e.* LM + Quant + SPI + HDR + CHDR.

Data	n	p	PIBF-CV	PIBF-OOB	$\widehat{PI}_\alpha$	OOB	RFPI	SC	QRF	GRF
Abalone	4177	8	102.37	64.39	17.65	5.20	422.10	20.42	45.39	6.32
Air quality (AH)	6941	13	383.60	126.95	26.63	11.55	1256.56	45.45	126.36	16.07
Air quality (RH)			383.53	124.71	26.60	11.55	1111.16	46.13	127.02	15.95
Airfoil selfnoise	1503	5	256.92	16.61	3.57	0.31	271.35	1.80	3.65	0.65
Ames housing	2929	80	195.07	28.56	15.64	8.73	333.28	42.95	226.49	11.34
Auto MPG	392	7	11.02	7.08	1.82	0.35	47.84	0.58	1.73	0.40
Boston housing	506	13	14.77	8.89	2.32	0.49	63.20	1.31	3.63	0.70
Computer hardware	209	6	6.53	3.80	0.93	0.22	26.49	0.28	0.98	0.24
Concrete compression	1030	8	28.28	17.80	3.38	0.38	257.19	2.03	5.86	0.66
Concrete slump	103	7	9.52	2.10	0.65	0.14	26.60	0.13	0.62	0.11
Energy efficiency (CL)	768	8	85.97	15.93	2.51	0.34	341.73	0.94	2.39	0.52
Energy efficiency (HL)			99.95	20.31	2.47	0.34	380.97	0.92	2.42	0.42
Servo	167	4	12.05	3.99	0.72	0.15	53.14	0.12	0.65	0.15

global predictive performance. Hence, ensuring that the sample coverage level is achieved should be sufficient for most applications. The proposed calibration method with cross-validation is designed to ensure the sample coverage property. From the simulation study and real data analyses, we can see that the sample coverage is attained with the proposed calibration method.

5 Availability

The package is available from CRAN at https://cran.r-project.org/package=RFpredInterval. The development version is available at https://github.com/calakus/RFpredInterval.

6 Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by Fondation HEC Montréal.

Bibliography

C. Alakus, D. Larocque, and A. Labbe. *RFpredInterval: Prediction Intervals with Random Forests and Boosted Forests*, 2022. URL https://cran.r-project.org/package=RFpredInterval. R package version 1.0.5. [p302]

C. Alakuş, D. Larocque, S. Jacquemont, F. Barlaam, C.-O. Martin, K. Agbogba, S. Lippé, and A. Labbe. Conditional canonical correlation estimation based on covariates with random forests. *Bioinformatics*, 2021. URL https://doi.org/10.1093/bioinformatics/btab158. [p300, 303]

S. Athey, J. Tibshirani, and S. Wager. Generalized random forests. *The Annals of Statistics*, 47(2): 1148–1178, Apr. 2019. URL https://doi.org/10.1214/18-AOS1709. [p300, 301, 310]

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug. 1996. URL https://doi.org/10.1007/BF00058655. [p308]

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001. URL https://doi.org/10.1023/A:1010933404324. [p300, 301]

L. Breiman and L. Breiman. *Classification and regression trees*. The Wadsworth & Brooks/Cole statistics/probability series. Wadsworth International Group, Belmont, Calif., 1984. ISBN 0-534-98054-6 978-0-534-98054-2. [p301]

D. De Cock. Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. *Journal of Statistics Education*, 19(3), Nov. 2011. URL https://doi.org/10.1080/10691898.2011.11889627. [p300, 305]

D. Dua and C. Graff. UCI machine learning repository, 2017. URL https://archive.ics.uci.edu/ml. [p313]

I. Ghosal and G. Hooker. Boosting Random Forests to Reduce Bias; One-Step Boosted Forest and Its Variance Estimate. *Journal of Computational and Graphical Statistics*, 30(2):493–502, Apr. 2021. URL https://doi.org/10.1080/10618600.2020.1820345. [p301, 302]

T. Hothorn and A. Zeileis. Predictive Distribution Modeling Using Transformation Forests. *Journal of Computational and Graphical Statistics*, pages 1–16, Jan. 2021. URL https://doi.org/10.1080/10618600.2021.1872581. [p301, 309]

T. Hothorn, B. Lausen, A. Benner, and M. Radespiel-Tröger. Bagging survival trees. *Statistics in Medicine*, 23(1):77–91, 2004. URL https://doi.org/10.1002/sim.1593. [p300, 301, 303]

H. Ishwaran and U. B. Kogalur. *randomForestSRC: Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC)*, 2021. URL https://cran.r-project.org/package=randomForestSRC. R package version 2.11.0. [p304]

Jerome H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67, Mar. 1991. URL https://doi.org/10.1214/aos/1176347963. [p308]

M. Kuhn. *AmesHousing: The Ames Iowa Housing Data*, 2020. URL https://cran.r-project.org/package=AmesHousing. R package version 0.0.4. [p305]

J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman. Distribution-Free Predictive Inference for Regression. *Journal of the American Statistical Association*, 113(523):1094–1111, July 2018. URL https://doi.org/10.1080/01621459.2017.1307116. [p301]

F. Leisch and E. Dimitriadou. *mlbench: Machine Learning Benchmark Problems*, 2021. URL https://cran.r-project.org/package=mlbench. R package version 2.1-3. [p308]

Y. Lin and Y. Jeon. Random Forests and Adaptive Nearest Neighbors. *Journal of the American Statistical Association*, 101(474):578–590, June 2006. URL https://doi.org/10.1198/016214505000001230. [p300, 303]

B. Lu and J. Hardin. *forestError: A Unified Framework for Random Forest Prediction Error Estimation*, 2020. URL https://cran.r-project.org/package=forestError. R package version 0.2.0. [p302]

B. Lu and J. Hardin. A Unified Framework for Random Forest Prediction Error Estimation. *Journal of Machine Learning Research*, 22(8):1–41, 2021. URL https://jmlr.org/papers/v22/18-558.html. [p302, 303]

A. Mayr, T. Hothorn, and N. Fenske. Prediction intervals for future BMI values of individual children - a non-parametric approach by quantile boosting. *BMC Medical Research Methodology*, 12(1):6, Jan. 2012. URL https://doi.org/10.1186/1471-2288-12-6. [p315]

N. Meinshausen. Quantile Regression Forests. *Journal of Machine Learning Research*, 7(35):983–999, 2006. URL https://jmlr.org/papers/v7/meinshausen06a.html. [p301]

N. Meinshausen. *quantregForest: Quantile Regression Forests*, 2017. URL https://cran.r-project.org/package=quantregForest. R package version 1.3-7. [p301]

L. Mentch and G. Hooker. Quantifying Uncertainty in Random Forests via Confidence Intervals and Hypothesis Tests. *Journal of Machine Learning Research*, 17(26):1–41, 2016. URL https://jmlr.org/papers/v17/14-168.html. [p301]

H. Moradian, D. Larocque, and F. Bellavance. L1 splitting rules in survival forests. *Lifetime Data Analysis*, 23(4):671–691, Oct. 2017. URL https://doi.org/10.1007/s10985-016-9372-1. [p300, 301, 303]

H. Moradian, D. Larocque, and F. Bellavance. Survival forests for data with dependent censoring. *Statistical Methods in Medical Research*, 28(2):445–461, Feb. 2019. URL https://doi.org/10.1177/0962280217727314. [p300, 301, 303]

M.-H. Roy and D. Larocque. Prediction intervals with random forests. *Statistical Methods in Medical Research*, 29(1):205–229, Jan. 2020. URL https://doi.org/10.1177/0962280219829885. [p300, 301, 302, 303, 304, 305, 306, 309, 310, 311, 314, 315]

S. Tabib and D. Larocque. Non-parametric individual treatment effect estimation for survival data with random forests. *Bioinformatics*, 36(2):629–636, Jan. 2020. URL https://doi.org/10.1093/bioinformatics/btz602. [p300, 303]

J. Tibshirani, S. Athey, E. Sverdrup, and S. Wager. *grf: Generalized Random Forests*, 2021. URL https://cran.r-project.org/package=grf. R package version 2.0.2. [p301]

R. Tibshirani. *conformalInference: Tools for conformal inference in regression*, 2019. URL https://github.com/ryantibs/conformal. R package version 1.1. [p301]

V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. Springer Science & Business Media, Dec. 2005. ISBN 978-0-387-25061-8. [p301]

V. Vovk, I. Nouretdinov, and A. Gammerman. On-line predictive linear regression. *The Annals of Statistics*, 37(3):1566–1590, June 2009. URL https://doi.org/10.1214/08-AOS622. [p301]

S. Wager and S. Athey. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association*, 113(523):1228–1242, July 2018. URL https://doi.org/10.1080/01621459.2017.1319839. [p301]

M. N. Wright, S. Wager, and P. Probst. *ranger: A Fast Implementation of Random Forests*, 2020. URL https://cran.r-project.org/package=ranger. R package version 0.12.1. [p304]

G. Zhang and Y. Lu. Bias-corrected random forests in regression. *Journal of Applied Statistics*, 39(1):151–160, Jan. 2012. URL https://doi.org/10.1080/02664763.2011.578621. [p301]

H. Zhang. *rfinterval: Predictive Inference for Random Forests*, 2019. URL https://cran.r-project.org/package=rfinterval. R package version 1.0.0. [p302]

H. Zhang, J. Zimmerman, D. Nettleton, and D. J. Nordman. Random Forest Prediction Intervals. *The American Statistician*, 74(4):392–406, Oct. 2020. URL https://doi.org/10.1080/00031305.2019.1585288. [p301, 302]

Cansu Alakus
Department of Decision Sciences
HEC Montréal
Canada
(ORCiD: 0000-0001-7669-5863)
cansu.alakus@hec.ca

Denis Larocque
Department of Decision Sciences
HEC Montréal
Canada
(ORCiD: 0000-0002-7372-7943)
denis.larocque@hec.ca

Aurélie Labbe
Department of Decision Sciences
HEC Montréal
Canada
aurelie.labbe@hec.ca

etrm: Energy Trading and Risk Management in R

by Anders D. Sleire

Abstract This paper introduces **etrm**, an R package with tools for trading and financial risk management in energy markets. Contracts for electric power and natural gas differ from most other commodities due to the fact that physical delivery takes place over a time interval, and not at a specific point in time. There is typically strong seasonality, limited storage and transmission capacity and strong correlation between price and required volume. Such characteristics need to be taken into account when pricing contracts and managing financial risk related to energy procurement. Tools for these task are usually bundled into proprietary Energy Trading Risk Management (ETRM) systems delivered by specialized IT vendors. The **etrm** package offers a transparent solution for building a forward price curve for energy commodities which is consistent with methods widely used in the industry. The user's fundamental market view may be combined with contract price quotes to form a forward curve that replicate current market prices, as described in Ollmar (2003) and Benth et al. (2007b). **etrm** also provides implementations of five portfolio insurance trading strategies for energy price risk management. The forward market curve and the energy price hedging strategies are core elements in an ETRM system, which to the best of the author's knowledge has not been previously available in the R ecosystem.

1 Introduction

The purpose of this paper is to introduce the R package **etrm** and its tools for energy trading and financial risk management. Substantial fluctuations in energy prices represent a significant risk for market players, in particular for large consumers, producers and utility companies, see Benini et al. (2002). The price dynamics is complex due to strong weather dependency and physical constraints related to storage, distribution, and the introduction of new technology. See for example Nicolosi (2010) for an analysis of renewable energy production and the negative prices following extreme events in the German power market. Derivatives securities, such as futures contracts, are often used to hedge against the commodity price risk. Specialized Energy Trading and Risk Management (ETRM) systems provide the necessary tools to handle key activities such as position management, valuation and risk reporting. Several proprietary alternatives exist. The annual *Energy Risk's Software Survey* in Farrington (2020) gives an overview of major providers along with rankings based on industry polls. There, ETRM systems are divided into the operational categories *derivatives software*, *physical trading and operations software* and *front- and middle-office functionality*. Historically, many system providers within this domain have bundled modules into large monolithic architectures serving a variety of purposes, including accounting and regulatory reporting. During the last decade, a general trend within system development has moved towards splitting software into smaller stand-alone components. The **etrm** package solely focus on financial trading, and may be viewed as a module for front- and middle-office functionality for energy derivatives. The package currently offers transparent tools for two main ETRM activities 1) construction of forward market curves and 2) implementation of trading strategies for price risk management.

After the liberalization of electricity and gas markets started in the 1990s, a rich research literature emerged. Topics studied include pricing and hedging in the forward market and modelling of spot price processes, see for example Bessembinder and Lemmon (2002), Janczura et al. (2013) and Benth et al. (2007a). Alternative methods for pricing options in power markets can be found in Burger et al. (2004) and Benth and Schmeck (2014). Textbooks such as Eydeland and Wolyniec (2002), Benth et al. (2008) and Kirschen and Strbac (2018) may be used to gain a more detailed overview of market structure, available instruments, methods for risk management and the related markets for fuel, freight and weather products. In this paper, we will cover some of the theory regarding forward curve modelling from Ollmar (2003) and Benth et al. (2007b). The theoretical framework for price risk management is gathered from the *portfolio insurance* literature, see Leland (1980), Perold and Sharpe (1988), Leland and Rubinstein (1976). This will be presented in further detail below.

We would like to note that there are some tools available outside the domain of proprietary ETRM software providers. Two examples are the MathWorks case studies Sundar (2021) and Deoras (2021), focusing on risk assessment for gas-fired power plants and electricity load and price forecasting using MATLAB. These topics are however somewhat ad-hoc, and the supplied code cannot be easily incorporated into a generic ETRM system for general use. Similarly within the R ecosystem, there are related tools in the Rmetrics suite of packages, such as **fOptions** and **fPortfolio**, but they are not

directly applicable. Due to the unique properties of energy markets, standard methods for generating forward curves in interest rate markets cannot be used either. To fill this gap and provide practitioners and researchers with tools dedicated to energy price risk management, we have created **etrm**. The package is available on CRAN, and may be installed and loaded into the R environment by running the following commands:

```
if(!require(etrm)==TRUE) {install.packages("etrm")}
library(etrm)
```

The rest of the article is organized as follows. First, we give a brief introduction to energy market forward curves and the maximum smoothness forward curve (MSFC) model. We describe the **etrm** implementation and provide examples of use. Second, a short treatment of energy price risk management with futures contracts is provided, followed by a presentation of five portfolio insurance models. The **etrm** implementation is described and illustrated with practical examples for energy portfolios with both short and long market exposure. The third part provide an overview of the **etrm** package structure, available functions and included data sets. Finally, the last section summarizes the paper and provide some suggestions for future work.

2 Energy market forward price curves

The standardised forwards for electricity and gas are contracts for flow delivery. The underlying commodity is not received at a fixed point in time, but over a time interval. In mature markets, participants can trade a variety of products, both over-the-counter (OTC) and on exchanges such as Nasdaq Commodities, European Energy Exchange and the Intercontinental Exchange. Liquidity is often best in the so called front-products, and there is normally higher activity in contracts for next week, month, quarter and year compared to similar products further ahead in time. Shorter period contracts may not even be available on a longer horizon, and seasonal price variation is thus not directly observable in prices far ahead in time. Transacted volume and prices also inhibit quite pronounced seasonality, during the year, week and within a specific day. For this reason, forward contracts are divided into categories based on a *load pattern*. In the *base load* contracts, volume is delivered at a constant rate during the contract period, while *peak load* products are linked to high volume hours, such as Monday to Friday from 8 am to 8 pm. Other, more exotic load patterns exist, but they are less common. Further details can be found in Eydeland and Wolyniec (2002) and Kirschen and Strbac (2018).

The aim of the energy market forward price curve calculation is to create a compact representation of the forward market, at a given point in time. The curve must be able to price the quoted instruments correctly, while accounting for typical energy market characteristics such as seasonality and (possibly overlapping) contracts for flow delivery. The curve is an essential decision making tool with many uses, such as pricing non-standard supply agreements, investment decisions and risk management.

The topic of forward curve fitting has been studied for decades in interest rate markets, see for example McCulloch (1971) and Anderson et al. (1996). These techniques cannot be applied directly to commodities with flow delivery and strong seasonality in prices. There are several alternative approaches to calculating a forward price curve for energy commodities. In Fleten and Lemming (2003), market data is combined with forecasts generated by a bottom-up model constrained by the bid/ask spread in order to meet the no-arbitrage condition. Borak and Weron (2008) propose a semiparametric factor model for the forward curve dynamics in electricity markets, while Hildmann et al. (2012) develop a calculation method by combining parametric estimation and prediction of futures prices under constraints.

In **etrm**, we have opted for a method that combines a seasonal function with the maximum smoothness-approach from interest rate markets, see Adams and Van Deventer (1994). Hence, we follow the approach in Ollmar (2003) and Benth et al. (2007b). *Base load* contracts are used to calculate a curve with daily granularity. This method has several benefits. It produces a continuous curve which has a closed form solution, it is fast to calculate, flexible and used by many practitioners in the industry. The next section provide a brief overview of the method, followed by a description of the **etrm** implementation with examples.

Maximum smoothness forward curve model

Consider a market at time t with m forward contracts available for trading. Let the list

$$S_t = \{(\tau_1^s, \tau_1^e), (\tau_2^s, \tau_2^e), ..., (\tau_m^s, \tau_m^e)\}$$

contain the start and end dates for each of these contracts. The time distance between τ_i^s and τ_i^e for a contract i in $1, .., m$ cover standardized periods such as *week*, *month*, *quarter* and *year*. Some of these settlement intervals might overlap, and in order to handle this we create a new list of dates $\{t_0, t_1, ..., t_n\}$ to identify each separate sub period, see Figure 1. The new list is made by sorting all dates in S_t in ascending order and removing duplicates.

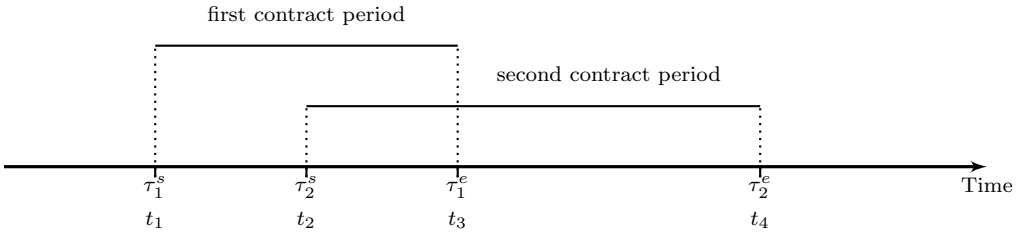


Figure 1: Illustration of two overlapping contracts with start (τ^s) and end (τ^e) dates. Due to the overlap, the total delivery period is split into sub intervals identified with $\{t_1, t_2, t_3, t_4\}$.

The forward price at time t for one unit of energy delivered at a constant rate $(\tau^e - \tau^s)^{-1}$ over the time interval (τ^s, τ^e) is denoted by $F(t, \tau^s, \tau^e)$, where $t \leq \tau^s < \tau^e$. A forward contract for a flow delivery may be thought of as the average of hypothetical single-delivery contracts. At time t, each of these would have a unique price $f(t, u)$ for the delivery at u with an infinitesimal delivery period. This leads to $F(t, \tau^s, \tau^e)$ being the weighted average

$$F(t, \tau_s, \tau_e) = \int_{\tau^s}^{\tau^e} w(u, \tau_s, \tau_e) f(t, u) \, \mathrm{d}u \tag{1}$$

where $w(u, \tau_s, \tau_e) = \frac{\hat{w}(u)}{\int_{\tau_s}^{\tau_e} \hat{w}(v) dv}$ is a weight function accounting for the rate of interest r and the time value of money. If the contract in question is settled at the end of the delivery period (forward contract), the weight function is given by $w(u, \tau_s, \tau_e) = 1/(\tau_e - \tau_s)$. If the contract is settled continuously over the delivery period (futures contract), $w(u, \tau_s, \tau_e) = \frac{re^{-ru}}{e^{-r\tau_s} - e^{-r\tau_e}}$. In the following we construct a forward market price curve for the entire horizon using a simplified notation $f(u)$ for the function describing the forward curve at time t. In order to model the strong seasonality in energy markets, the forward curve function is decomposed into two elements:

$$f(u) = \Lambda(u) + \epsilon(u) \qquad u \in [t_0, t_n] \tag{2}$$

Following Ollmar (2003) we calculate $f(u)$ by combining a prior function $\Lambda(u)$ which contain our subjective views on the future prices with an adjustment function $\epsilon(u)$ to ensure match with the observed closing prices for the m contracts. The prior could be generated with a simple sinusoidal function or from a fundamental model more capable of describing the seasonality and calendar effects observed in energy markets. Should the prior be excluded, the seasonal price patterns will not be visible in the far end of the curve, where only yearly or seasonal contracts are available. Smoothing is calculated on the adjustment function, we aim to minimize the total curvature of $\Lambda(u)$ while preserving the information from the prior. Smoothness is defined as the integral of the second-order derivative of the function, and the smoothest possible curve over $[t_0, t_n]$ is achieved by minimising

$$\int_{t_0}^{t_n} [\epsilon''(u)]^2 \, \mathrm{d}u$$

under five constraints presented below. Lim and Xiao (2002) show the smoothest possible curve is found when the n sub periods are modelled by fourth-degree polynomials. We write $\epsilon(u)$ as a spline

$$\epsilon(u) = \begin{cases} a_1 u^4 + b_1 u^3 + c_1 u^2 + d_1 u + e_1, & u \in [t_0, t_1], \\ a_2 u^4 + b_2 u^3 + c_2 u^2 + d_2 u + e_2, & u \in [t_1, t_2], \\ \quad . & \\ \quad . & \\ a_n u^4 + b_n u^3 + c_n u^2 + d_n u + e_n, & u \in [t_{n-1}, t_n]. \end{cases}$$

In order to construct the forward curve function we need to identify the parameters of $\epsilon(u)$

$$\mathbf{x}^\intercal = [a_1\ b_1\ c_1\ d_1\ e_1 \quad a_2\ b_2\ c_2\ d_2\ e_2 \ldots\ a_n\ b_n\ c_n\ d_n\ e_n]$$

by solving the quadratic optimisation problem

$$\min_x \int_{\tau^s}^{\tau^e} [\epsilon''(u;x)]^2\,\mathrm{d}u \tag{3}$$

subject to

$$\begin{aligned}
(a_{j+1}-a_j)u_j^4 + (b_{j+1}-b_j)u_j^3 + (c_{j+1}-c_j)u_j^2 + (d_{j+1}-d_j)u_j + e_{j+1} - e_j &= 0 \quad \text{(a)}\\
4(a_{j+1}-a_j)u_j^3 + 3(b_{j+1}-b_j)u_j^2 + 2(c_{j+1}-c_j)u_j + (d_{j+1}-d_j) &= 0 \quad \text{(b)}\\
12(a_{j+1}-a_j)u_j^2 + 6(b_{j+1}-b_j)u_j + 2(c_{j+1}-c_j) &= 0 \quad \text{(c)}\\
\epsilon'(u_n;x) &= 0 \quad \text{(d)}\\
\frac{1}{\tau_i^e-\tau_i^s}\int_{\tau_i^s}^{\tau_i^e} (\Lambda(u)+\epsilon(u))\,\mathrm{d}u &= F_i^c \quad \text{(e)}
\end{aligned}$$

for spline knot $j = 1, ..., n-1$ and contract $i = 1, ..., m$. The constraint in (a) ensures the adjustment function is continuous in the knots, while (b) and (c) imposes this restriction also for the first and second order differentials. The (d) constraint require the adjustment function to be horizontal at time T, and finally (e) also require the average value of the forward price function $f(u)$ over the delivery period for contract i to match the quoted closing price F_i^c. Here, we could take the interest rate effect from r into account and set the present value of the average of the forward price function equal to present value of the forward contract. Instead of doing that, we will follow Benth et al. (2008) and Ollmar (2003) and assume $r = 0$ such that the weight function in 1 is approximated with $w(u, \tau^s, \tau^e) \approx \frac{1}{\tau^e-\tau^s}$. Like Benth et al. (2008) we will argue that both the prior and the smoothing will outweigh a marginal interest rate effect. This minimisation problem can be expressed as

$$\min_{\mathbf{x}} \quad \mathbf{x}^\intercal \mathbf{H} \mathbf{x},$$

where $\mathbf{x}$ is a $(5n \times 1)$ vector and

$$\mathbf{H} = \begin{bmatrix} h_1 & & 0 \\ & \ddots & \\ 0 & & h_n \end{bmatrix}, \mathbf{h}_j = \begin{bmatrix} \frac{144}{5}\Delta_j^5 & 18\Delta_j^4 & 8\Delta_j^3 & 0 & 0 \\ 18\Delta_j^4 & 12\Delta_j^3 & 6\Delta_j^2 & 0 & 0 \\ 8\Delta_j^3 & 6\Delta_j^2 & 4\Delta_j & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The block diagonal matrix $\mathbf{H}$ has dimensions $(5n \times 5n)$ and $\Delta_j = t_{j+1}^l - t_j^l$. As the constraints in (3) are all linear in x and may be expressed on the form $\boldsymbol{Ax} = \boldsymbol{B}$, the problem may be rephrased as an unconstrained minimisation problem via the Lagrange multiplier method:

$$\min_{\mathbf{x},\boldsymbol{\lambda}} \quad \boldsymbol{x}^\intercal \boldsymbol{H}\boldsymbol{x} + \boldsymbol{\lambda}^\intercal(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{B})$$

where $\mathbf{A}$ is a $(3n + m - 2 \times 5n)$ matrix and $\mathbf{B}$ is a $(3n + m - 2 \times 1)$ vector. The spline parameter vector and Lagrange multipliers are identified by solving

$$\begin{bmatrix} \mathbf{2H} & \mathbf{A}^\intercal \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B} \end{bmatrix} \tag{4}$$

where the left matrix has dimension of $(8n + m - 2) \times (8n + m - 2)$ and both vectors are of dimension $(8n + m - 2)$.

The "MSFC" class with examples

The forward curve calculation in **etrm** is implemented in the S4 `"MSFC"` class. By supplying all required arguments to the constructor function `msfc()`, the user may create an object that contains the calculation results, input arguments and further calculation details. In addition to the arguments from the list of contracts, the user may also provide a prior to the calculation. By default the prior is set to zero, but the user can input any vector expressing a belief regarding the market to be combined with the observed prices. An overview of the `msfc()` arguments can be found in Table 1.

Argument	Description	Default value
`tdate`	Trading date	none
`include`	Logical vector for contract selection	none
`contract`	Character vector with contract names	none
`sdate`	Date vector with contract start dates	none
`edate`	Date vector with contract end dates	none
`f`	Numeric vector with contract prices	none
`prior`	Numeric prior curve vector	0

Table 1: Arguments for the `msfc()` constructor function for forward curve calculation.

The `"MSFC"` class properties and available methods are summarised in Figure 2, and a brief description is provided in the following. The `"Name"` slot describe the type of forward curve model used by storing the character value "MSFC", while `"TradeDate"` keeps the trade date used in the calculation. A data frame containing details for selected contracts along with the calculated forward price based on the curve can be found in `"BenchSheet"`. A count of the $(n-1)$ number of polynomials used in the spline is available as a scalar value in `"Polynomials"`, and the prior curve vector in `"PriorFunc"`. The main calculation result is stored in a data frame which contains daily values for all selected contracts along with the calculated forward curve. The data frame span the date range from `"TradeDate"` to the end date of the contract furthest ahead in time, and can be found in `"Results"`. The interested user may also extract additional information regarding the spline itself. Coefficients for all polynomials can be found in the `"SplineCoef"` list and the knotpoints separating them in the numeric vector `"KnotPoints"`. Further details regarding the calculation of the daily forward curve values are available in the `"CalcDat"` data frame. This table is essentially an extended version of `"Results"`, where numeric time vectors and the spline coefficients have been added.

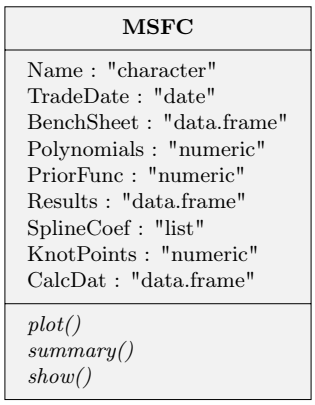


Figure 2: Attributes and methods of the `"MSFC"` class.

The `"MSFC"` class has the generic methods `plot()`, `summary()` and `show()`. The `plot()` method may be used to create a chart of the calculated curve and underlying contracts from the `"Results"` data frame. All plot methods in **etrm** are based on **ggplot2**, see Wickham (2011a). The `summary()` method returns a list with three elements; a description string, a sample of the prior vector, and the bench sheet. Finally, the `show()` method returns the `"Results"` data frame.

We proceed with a practical example using two of the embedded **etrm** data sets to represent information available to a European power market participant. All market-related inputs to the `msfc()` constructor (trade date and contract properties) are required arguments. These are collected from a synthetic data set for the trading date 2013-05-13, and can be found in `"powfutures130513"` presented in Table 2. Contracts covering long time spans are excluded with `include = FALSE` if futures of shorter duration are available for the same time interval in order to preserve the seasonality available in market prices. We use a seasonal prior with high energy prices during the winter season, followed by

a drop toward the lower summer levels. It also take into account some well known calendar effects, such as weekends. The prior is simple, and merely used for illustrative purposes. It can be found in the data set "powpriors130513" included in the package. A calculation excluding the prior is also added for comparison.

Include	Contract	Start	End	Closing
TRUE	W21-13	2013-05-20	2013-05-26	33.65
TRUE	W22-13	2013-05-27	2013-06-02	35.77
TRUE	W23-13	2013-06-03	2013-06-09	36.58
TRUE	W24-13	2013-06-10	2013-06-16	35.93
TRUE	W25-13	2013-06-17	2013-06-23	33.14
TRUE	W26-13	2013-06-24	2013-06-30	34.16
FALSE	MJUN-13	2013-06-01	2013-06-30	35.35
TRUE	MJUL-13	2013-07-01	2013-07-31	33.14
TRUE	MAUG-13	2013-08-01	2013-08-31	35.72
TRUE	MSEP-13	2013-09-01	2013-09-30	38.41
TRUE	MOCT-13	2013-10-01	2013-10-31	38.81
TRUE	MNOV-13	2013-11-01	2013-11-30	40.94
FALSE	Q3-13	2013-07-01	2013-09-30	35.72
TRUE	Q4-13	2013-10-01	2013-12-31	40.53
TRUE	Q1-14	2014-01-01	2014-03-31	42.40
TRUE	Q2-14	2014-04-01	2014-06-30	33.39
TRUE	Q3-14	2014-07-01	2014-09-30	31.78
TRUE	Q4-14	2014-10-01	2014-12-31	38.25
TRUE	Q1-15	2015-01-01	2015-03-31	40.73
TRUE	Q2-15	2015-04-01	2015-06-30	32.64
TRUE	Q3-15	2015-07-01	2015-09-30	30.87
TRUE	Q4-15	2015-10-01	2015-12-31	37.22
FALSE	CAL-14	2014-01-01	2014-12-31	36.43
FALSE	CAL-15	2015-01-01	2015-12-31	35.12
TRUE	CAL-16	2016-01-01	2016-12-31	34.10
FALSE	CAL-17	2017-01-01	2017-12-31	35.22
FALSE	CAL-18	2018-01-01	2018-12-31	36.36

Table 2: Closing prices for futures contracts used in the forward curve calculation for 2013-05-13. Contracts are selected for the calculations with the `include` vector. Prices for these contracts can be found as horizontal lines in Figure 3

As shown in Figure 3, the shorter contracts close in time to the trading date clearly reflect a seasonal pattern. This is typical in power markets, where weather and calendar effects have strong influence on transacted volume and price formation. On a longer horizon however, this information is not observable in market prices, as the quoted contracts cover longer time spans. This is where price data may be supplemented with prior knowledge in order to create a representation of the market consistent with both the underlying fundamentals and the listed contracts. The following code will create the "MSFC" objects and plot calculation results:

```
library(etrm)
library(gridExtra)
data(powfutures130513)
data(powpriors130513)

# instance of MSFC class with prior
fwd.fut.wpri <- msfc(tdate = as.Date("2013-05-13"),
                     include = powfutures130513$Include,
                     contract = powfutures130513$Contract,
                     sdate = powfutures130513$Start,
                     edate = powfutures130513$End,
                     f = powfutures130513$Closing,
                     prior = powpriors130513$mod.prior)

# instance of MSFC class without prior
fwd.fut.npri <- msfc(tdate = as.Date("2013-05-13"),
                     include = powfutures130513$Include,
                     contract = powfutures130513$Contract,
```

```
                    sdate = powfutures130513$Start,
                    edate = powfutures130513$End,
                    f = powfutures130513$Closing,
                    prior = 0)
```

```
# the generic plot() method
pw <- plot(fwd.fut.wpri, ylab = "EUR/MWh", legend = "")
pn <- plot(fwd.fut.npri, ylab = "EUR/MWh", legend = "")

# combine plots
gridExtra::grid.arrange(pw, pn)
```

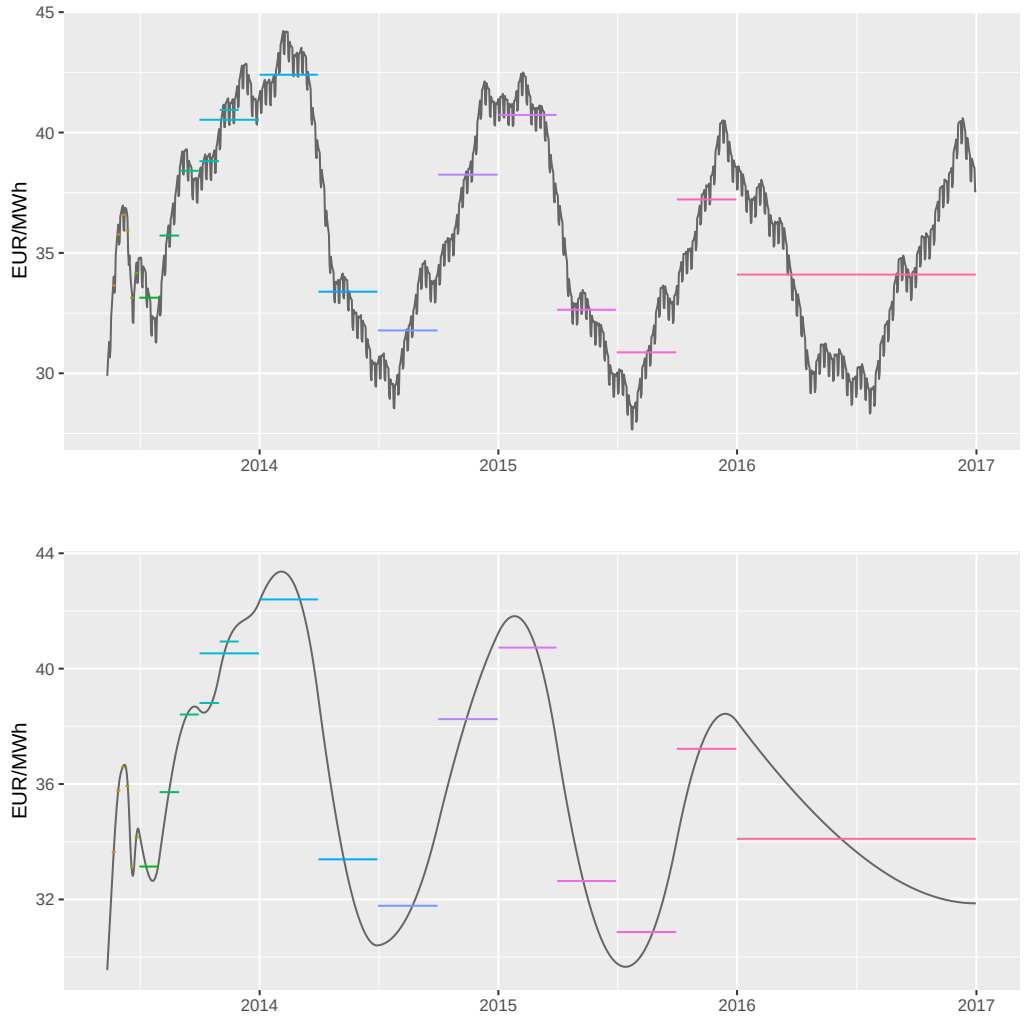


Figure 3: Two alternative forward curve calculations based on the same contract selection. In the top panel, a prior function is included in the calculation. This curve shows price variation on the weekly level, with lower prices during weekends. The prior also ensures that yearly seasonality is visible in the far end of the curve. The bottom plot is based solely on market prices, which does not reflect seasonality on such long horizon.

The computed prices may be verified via the `summary()` method, which also return a sample of the prior and information regarding the spline calculation:

```
> summary(fwd.fut.wpri)
$Description
[1] "MSFC of length 1329 built with 41 polynomials at trade date 2013-05-13"

$PriorFunc
[1] 30.10842 30.16396 30.19572 30.16144 29.06268 28.93272
```

```
$BenchSheet
   Include Contract       From         To Price  Comp
1     TRUE   W21-13 2013-05-20 2013-05-26 33.65 33.65
2     TRUE   W22-13 2013-05-27 2013-06-02 35.77 35.77
3     TRUE   W23-13 2013-06-03 2013-06-09 36.58 36.58
4     TRUE   W24-13 2013-06-10 2013-06-16 35.93 35.93
5     TRUE   W25-13 2013-06-17 2013-06-23 33.14 33.14
6     TRUE   W26-13 2013-06-24 2013-06-30 34.16 34.16
8     TRUE  MJUL-13 2013-07-01 2013-07-31 33.14 33.14
9     TRUE  MAUG-13 2013-08-01 2013-08-31 35.72 35.72
10    TRUE  MSEP-13 2013-09-01 2013-09-30 38.41 38.41
11    TRUE  MOCT-13 2013-10-01 2013-10-31 38.81 38.81
12    TRUE  MNOV-13 2013-11-01 2013-11-30 40.94 40.94
14    TRUE    Q4-13 2013-10-01 2013-12-31 40.53 40.53
15    TRUE    Q1-14 2014-01-01 2014-03-31 42.40 42.40
16    TRUE    Q2-14 2014-04-01 2014-06-30 33.39 33.39
17    TRUE    Q3-14 2014-07-01 2014-09-30 31.78 31.78
18    TRUE    Q4-14 2014-10-01 2014-12-31 38.25 38.25
19    TRUE    Q1-15 2015-01-01 2015-03-31 40.73 40.73
20    TRUE    Q2-15 2015-04-01 2015-06-30 32.64 32.64
21    TRUE    Q3-15 2015-07-01 2015-09-30 30.87 30.87
22    TRUE    Q4-15 2015-10-01 2015-12-31 37.22 37.22
25    TRUE   CAL-16 2016-01-01 2016-12-31 34.10 34.10
```

The forward curve values can be extracted along with daily prices for the contracts used in the calculation with the `show()` method:

```
> head(show(fwd.fut.wpri), 20)[1:5]
         Date     MSFC W21-13 W22-13 W23-13
1  2013-05-13 29.89373     NA     NA     NA
2  2013-05-14 30.40235     NA     NA     NA
3  2013-05-15 30.88704     NA     NA     NA
4  2013-05-16 31.30634     NA     NA     NA
5  2013-05-17 30.66200     NA     NA     NA
6  2013-05-18 30.98687     NA     NA     NA
7  2013-05-19 32.33591     NA     NA     NA
8  2013-05-20 32.74655  33.65     NA     NA
9  2013-05-21 33.19772  33.65     NA     NA
10 2013-05-22 33.63844  33.65     NA     NA
11 2013-05-23 34.02161  33.65     NA     NA
12 2013-05-24 33.34168  33.65     NA     NA
13 2013-05-25 33.62327  33.65     NA     NA
14 2013-05-26 34.91272  33.65     NA     NA
15 2013-05-27 35.24208     NA  35.77     NA
16 2013-05-28 35.59669     NA  35.77     NA
17 2013-05-29 35.92499     NA  35.77     NA
18 2013-05-30 36.17633     NA  35.77     NA
19 2013-05-31 35.34194     NA  35.77     NA
```

```
20 2013-06-01 35.44437     NA  35.77     NA
```

We have excluded columns from the data frame for the sake of presentation. Further details regarding the calculation such as spline coefficients and knot points can be found in the slots:

```
> slotNames(fwd.fut.wpri)
[1] "Name"        "TradeDate"   "BenchSheet"
[4] "Polynomials" "PriorFunc"   "Results"
[7] "SplineCoef"  "KnotPoints"  "CalcDat"
```

See for example the numeric vector with knot points, measured in years from the trading date:

```
> fwd.fut.npri@KnotPoints
 [1] 0.00000000 0.01917808 0.03561644 0.03835616 0.05479452 0.05753425 0.07397260
 [8] 0.07671233 0.09315068 0.09589041 0.11232877 0.11506849 0.13150685 0.13424658
[15] 0.21643836 0.21917808 0.30136986 0.30410959 0.38356164 0.38630137 0.46849315
[22] 0.47123288 0.55068493 0.63561644 0.63835616 0.88219178 0.88493151 1.13150685
[29] 1.13424658 1.38356164 1.38630137 1.63561644 1.63835616 1.88219178 1.88493151
[36] 2.13150685 2.13424658 2.38356164 2.38630137 2.63561644 2.63835616 3.63835616
```

The coefficients for the first polynomial in the adjustment function spline can be found with

```
> fwd.fut.npri@SplineCoef[[1]]
[1] -355585.14451   10911.10580     -78.47028     151.90713      29.54903
```

The most elaborate presentation of the curve calculation is available in `fwd.fut.npri@CalcDat`. This slot contains a data frame with all calculation details for each of the daily values returned by `msfc()`. It is not included here due to space requirements.

3 Energy price risk management

Energy market participants may be exposed to number of risk factors such as volume, profile and basis risk, counter party defaults, foreign exchange and market liquidity, to name a few. See for example Eydeland and Wolyniec (2002) and Kirschen and Strbac (2018) for a comprehensive treatment of the topic. The main focus in **etrm** is on the market price risk of the energy commodity. Consider the price risk associated with the constant base load volume q to be delivered over a future time interval (τ^s, τ^e). The risk can be mitigated by taking positions in the futures market during a trading period (t_0, T), which ends before the actual delivery of the energy takes place at $T < \tau^s$.

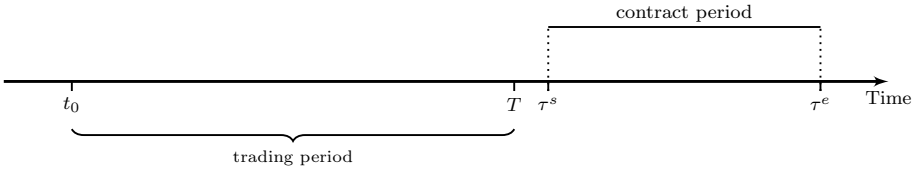


Figure 4: Trading and settlement periods for energy forward contract.

The price risk may be reduced by constructing a portfolio, consisting of the physical energy market exposure and derivatives contracts. The *portfolio price* per energy unit p_t is calculated as the weighted average of the value of the transacted volumes and the open volume evaluated mark-to-market

$$p_t = \frac{1}{q}\left(f_0 h_0 q + \sum_{i=1}^{t} f_i(h_i - h_{i-1})q + f_t(1-h_t)q\right) = f_0 h_0 + \sum_{i=1}^{t} f_i(h_i - h_{i-1}) + f_t(1-h_t) \qquad (5)$$

where $h_t \in (0,1)$ is the hedge rate and f_t the futures price at time t. In the simplest possible scenario, the risk can be managed by locking the entire volume in the forward market. This removes the price risk and the portfolio owner knows up front what to pay or receive when delivery of the energy takes place. On the downside, one might regret locking if the market develops in a favourable way.

Portfolio insurance strategies

In the portfolio insurance approach, dynamic hedging strategies that allow buying and selling the hedging instrument are used to protect the portfolio, while seeking to benefit from advantageous market developments. Historically, the theory of portfolio insurance has focused on protection against downside risk in financial investment portfolios, see Leland and Rubinstein (1976) , Perold and Sharpe (1988) and Leland (1980). Here, we apply the same principles to manage commodity price risk in the forward market. A consumer following a dynamic hedging program may control price risk by locking a share of future volume in the futures market, and increase (decrease) the share if the price increase (falls). A seller can implement similar strategies to maximise value of the energy portfolio. The size of the initial hedged share and how it is adjusted affects both the protective properties of the hedging scheme as well as its ability to exploit opportunities in the market. Trading activity needs

to be carefully managed and harmonized with overall objectives and risk preferences. A variety of portfolio insurance strategies offer different approaches to this task. The allocation strategies presented below all aim to control p_t and prevent breach of a pre-specified cap (or floor) price, p^*, under the hedge rate restriction $h_t \in (0,1)$.

The **Constant proportion portfolio insurance (CPPI)** strategy was introduced by Perold (1986) and Black and Jones (1987) for management of investment portfolios with capital guarantees. When applied to an energy portfolio, it aims to insure the portfolio by protecting a target price, a cap (floor) value for the portfolio price. Prior to start of hedging, p^* is set equal to the highest (lowest) acceptable portfolio price. This target price must be set higher than the first day's market price f_0 to implement cap protection, or lower for a floor protection model. The difference between the target price and the current portfolio price is termed the cushion. The key idea of CPPI is that the proportion of the volume exposed to the market should be calculated as a constant multiple m of the cushion. The multiple is given by $m = \mu^{-1}$, where $\mu > 0$ is a risk factor set to handle the maximum daily price change to be handled by the hedging model, which again affects strategy gearing. The unexposed proportion, the hedge rate, is

$$h_t = \begin{cases} 0 & \text{if} \quad c_t > \mu \\ 1 - mc_t & \text{if} \quad \mu \geq c_t \geq 0 \\ 1 & \text{if} \quad c_t < 0 \end{cases} \tag{6}$$

where $c_t = p^* - p_{t-1}$ for a short hedger, and $c_t = p_{t-1} - p^*$ for the long hedger.

In the **Dynamic proportion portfolio insurance (DPPI)** strategy, a decision rule similar to CPPI is applied, but the multiple m_t is allowed to vary. Changing market conditions may require re-evaluation of the risk factor in the multiple. Methods such as Value-at-Risk or Expexted Shortfall, or even simple heuristics can be used for this purpose. For a further treatment of DPPI type strategies, the reader is referred to Lee et al. (2008) and Chen et al. (2008). In **etrm** we also allow adjustments in the target price p_t^*, to catch opportunities to lower the capped value, or to increase the floor value. The hedge rate is determined similarly to CPPI, with

$$p_t^* = \begin{cases} min(\lambda p_{t-1}, p_{t-1}^*) & \text{short hedger} \\ \\ max(\lambda p_{t-1}, p_{t-1}^*) & \text{long hedger} \end{cases} \tag{7}$$

where $\lambda = \frac{p_o^*}{p_o}$ for a short hedger, and $\lambda = \frac{p_o}{p_o^*}$ for the long hedger.

Option based portfolio insurance (OBPI) was first introduced in Leland and Rubinstein (1976) as a means of providing insurance for investment portfolios. By combining an investment in a risky asset with a put option on the asset, the portfolio value is prevented from falling below the option strike price, K. A similar approach can be taken for the energy portfolio. As we are using futures contracts to manage the energy price risk, the Black-76 formula introduced in Black (1976) is used to approximate the contingent claim premium. The price at time t of the European call and put options with exercise date T and strike price K, on a futures contract with delivery start $\tau^s \geq T$ is given by

$$C(f_t, t, K, \sigma, r) = e^{-r(T-t)}[f_t N(d_1) - KN(d_2)] \tag{8}$$

$$P(f_t, t, K, \sigma, r) = e^{-r(T-t)}[KN(-d_2) - f_t N(-d_1)] \tag{9}$$

where f_t is the futures price and N is the cumulative distribution function of $N(0,1)$, where

$$d_1 = \frac{\ln(f_t/K) + (\sigma^2/2)(T-t)}{\sigma\sqrt{(T-t)}} \tag{10}$$

$$d_2 = d_1 - \sigma\sqrt{(T-t)} \tag{11}$$

and r is the risk free rate of interest, σ the volatility of the underlying futures price and $(T-t)$ is the time to exercise date. The sensitivity in the option premiums with respect to changes in the underlying futures price is given by the call and put option deltas:

$$\frac{\partial C}{\partial f} = e^{-r(T-t)}N(d_1) \tag{12}$$

$$\frac{\partial P}{\partial f} = e^{-r(T-t)}N(-d_1) \tag{13}$$

These are used to synthesise the option, by setting portfolio hedge rate $h_t \in (0,1)$ with the call (buyer) and put (seller) option deltas. By implementing this *delta hedging* scheme, a cap (floor) for the portfolio price is set at the option strike price K, adjusted for the option premium/ replication costs. For a more

detailed presentation of the underlying theory, the reader is referred to Bjork (2009).

Step hedge portfolio insurance (SHPI) is a simple and mechanical benchmark strategy that builds hedging positions gradually by transacting identical volumes each day through the trading period (t_0, T), reaching a full hedge prior to the start of the settlement period. The hedge rate for a buyer at time t is given by

$$h_t = \begin{cases} \frac{t}{T-t_0+1} & \text{if} \quad p_t < p^* \\ 1 & \text{if} \quad p_t \geq p^* \end{cases} \tag{14}$$

The hedges for a seller is entered mechanically in a similar manner as long as $p_t > p^*$. In the event $p_t \leq p^*$ a full hedge $h_t = 1$ is implemented. By distributing the transacted volumes evenly across the trading period while monitoring the target, the strategy portfolio price will either be locked in at the target price, or end up equal to the average forward market price over (t_0, T).

Finally, the **Stop loss portfolio insurance (SLPI)** is another simple benchmark, where no hedge positions are entered unless the target level is reached. For a buyer, this may be expressed as

$$h_t = \begin{cases} 0 & \text{if} \quad p_t < p^* \\ 1 & \text{if} \quad p_t \geq p^* \end{cases} \tag{15}$$

For a seller, the logic is reversed with $h_t = 0$ for $p_t > p^*$ and $h_t = 1$ for $p_t \leq p^*$. In the event that the target level is reached, the portfolio is kept fully hedged until start of settlement. If this does not occur, the portfolio follows the forward market, leaving an option to lock in the price at contract expiration.

The strategies presented above all have strengths and weaknesses. CPPI, SHPI and SLPI are simple and intuitive, but can be vulnerable to so-called *lock in*, the inability to improve portfolio price once the target level has been reached. This is also the case for DPPI. The OBPI does not suffer from this trait, but it relies on more assumptions regarding model parameters. In some scenarios, it will also generate more trading activity (costs), for example if the market fluctuate around the option strike price, K. Finally, as the strategies must be implemented in discrete time, they will all be exposed to gap risk.

The strategy classes with examples

The portfolio insurance strategies in **etrm** are implemented as S4 classes. Since they share many characteristics, they inherit most of their properties from a parent class, `"GenericStrat"`. In fact, the implementation of the simple benchmark strategies SLPI and SHPI do not require any additional properties to be added to the parent. The remaining strategy classes have some additional model specific features, in accordance with the descriptions in previous section. This modular design offers flexibility, and new strategies for price risk management can easily be added to the package.

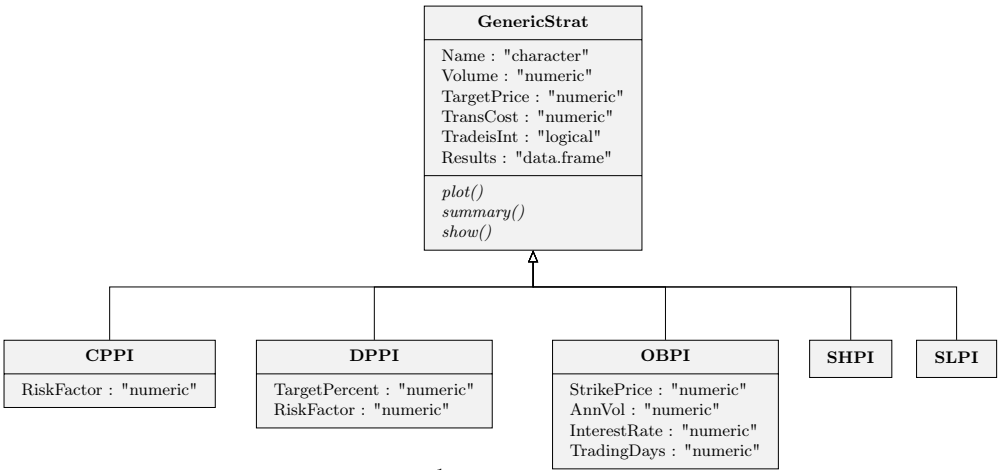


Figure 5: Attributes and methods for portfolio insurance strategy classes in the **etrm** package.

Figure 5 provide an overview of the class hierarchy, and a brief description is given in the following. In `"GenericStrat"`, the `"Name"` property is used to store a strategy identifier. Allowed character values are "CPPI", "DPPI", "OBPI", "SHPI" and "SLPI". The volume to be managed and the corresponding price cap (floor) can be found in `"Volume"` and `"TargetPrice"`, respectively. If a transaction cost has been included in the calculation of the portfolio price, this is to be found in `"TransCost"`. One may also set a restriction on transactions by requiring that the smallest volume available for trading is

equal to 1 unit. This lot size limitation is stored as TRUE/FALSE in `"TradeisInt"`. The main output from a strategy calculation can be found in `"Results"`. This data frame keeps daily values for market prices, transactions, exposed volume, open volume, hedge rate, target price and portfolio price.

The generic methods `plot()`, `summary()` and `show()` are implemented in `"GenericStrat"` and inherited by the strategy classes. The `plot()` method returns a chart based on `"Results"`, with daily values for portfolio, market and target prices and portfolio hedge rate. The `summary()` method returns a list with five elements; a description string, portfolio volume, target price, calculated churn rate and a data frame with summary statistics for the trading period. Finally, the `show()` method returns the `"Results"` data frame.

Argument	Description	Default value
`q`	Numeric volume	none
`tdate`	Date vector with trading days	none
`f`	Numeric price vector	none
`tcost`	Numeric transaction cost	0
`int`	Logical lot size integer restriction	TRUE

Table 3: Arguments shared by the portfolio insurance strategy functions.

The strategy constructor functions `cppi()`, `dppi()`, `obpi()`, `shpi()` and `slpi()` share five of the arguments, see Table 3. Each strategy require some additional arguments to implement the models presented in previous section. All of these inputs are of `"numeric"` data type. They are summarised in Table 4.

Function	Argument	Description	Default value
`cppi()`	`tper`	Target price factor	none
	`rper`	Risk factor percentage	none
`dppi()`	`tper`	Target price factor	none
	`rper`	Risk factor percentage	none
`obpi()`	`k`	Option strike price	$k = f_0$
	`vol`	Annualized volatility	none
	`r`	Interest rate	0
	`tdays`	Trading days per year	250
	`daysleft`	Days left to expiry	none
`shpi()`	`tper`	Target price factor	none
	`daysleft`	Days left to expiry	none
`slpi()`	`tper`	Target price factor	none

Table 4: Model specific arguments for the portfolio insurance strategy functions.

To illustrate further, we proceed with an example. Consider a European consumer of electricity procuring 30 MW to be delivered in 2006. The CAL-06 baseload power future from the synthetic **etrm** `"powcal"` data set is used as hedging instrument. Trading is started 500 days prior to the contract expiry, approximately a horizon of 2 years. For the `"OBPI"` strategy presented below, the target price is calculated as an expected price cap given by the option premium-adjusted strike price selected for the delta hedging scheme within a standard Black-76 option pricing framework. The default `obpi()` strike price is set at-the-money, in this case at 26.82 EUR/MWh. The expected target price illustrated with the horizontal dotted line in Figure 6 is calculated to be 29.84 EUR/MWh. The `"OBPI"` delta hedging scheme dictate an initial hedge rate of 57 percent, and the consumer enters a 17 MW position in CAL-06 on the first day of trading.

As time progresses and the market price changes, the `obpi()` function adjust the required hedge rate in order to replicate the call option on the CAL-06 contract. Hedge rate is gradually built up as the market increase from the second quarter of 2004, followed by a reduction after the sharp price drop starting late in the same year. Eventually, the volume is fully hedged due to the strong upwards price trend in 2005. The CAL-06 contract closes at 37.81 EUR/MWh on the expiry date, while the consumer has a hedge of 30 MW and a portfolio price of 29.29 EUR/MWh. The calculated price of the option to be synthesized (and the delta hedges) will depend on the Black-76 model parameters. In this example the risk free rate of interest is set to $r = 0$ and annualized volatility σ is assumed to be 20 percent. The following code will implement the strategy and create the plot in Figure 6:

```
# data frame with final 500 trading days for CAL-06 contract
dat06 <- tail(na.omit(powcal[, c(1,2)]), 500)

# instance of the OBPI class
cal06_obpi_b <- obpi(q = 30,
                     tdate = dat06$Date,
                     f = dat06$`CAL-06`,
                     k = dat06$`CAL-06`[1],
                     vol = 0.2,
                     r = 0,
                     tdays = 250,
                     daysleft = 500,
                     tcost = 0,
                     int = TRUE)

# the generic plot() method
plot(cal06_obpi_b, title = "", legend = "right", ylab.1 = "EUR/MWh")
```

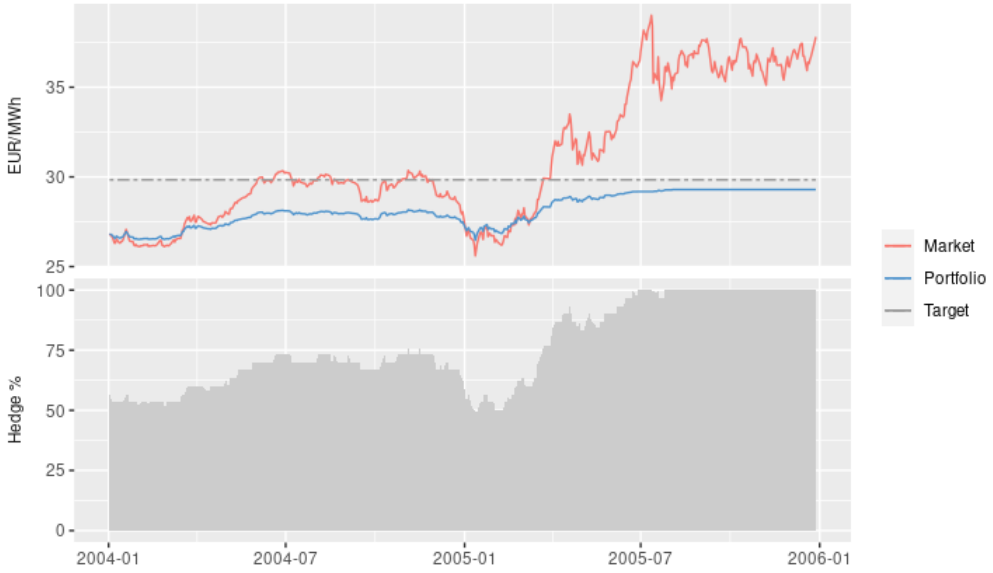


Figure 6: Option based portfolio insurance (OBPI) strategy for buyer CAL-06. Daily observations for prices (top panel) and hedge rate (bottom panel). As the market price continue to rise, the hedge rate is increased and the portfolio price is locked below the target price level.

An aggregated view of the trading activity over the 2 year period and final results can be retrieved by running the `summary()` method on the object created above:

```
> summary(cal06_obpi_b)
$Description
[1] "Hedging strategy of type OBPI and length 500"

$Volume
[1] 30

$Target
[1] 29.83626

$ChurnRate
[1] 4.333333
```

```
$Stats
      Market Trade Exposed Position     Hedge   Target Portfolio
First  26.82    17      13       17 0.5666667 29.83626  26.82000
Max    39.01    17      17       30 1.0000000 29.83626  29.29433
Min    25.60    -3       0       13 0.4333333 29.83626  26.46833
Last   37.81     0       0       30 1.0000000 29.83626  29.29433
```

We note from the "ChurnRate" that the underlying 30 MW volume had to be traded 4.33 times in order to synthesize the call option and achieve the results summarised in "Stats". By also considering the trading costs in the calculations, the user can get valuable inputs when considering alternatives, such as simply buying the option in the market. However, such contract may not always be available.

Finally, the show() method provide details regarding daily values for market price, transactions, exposed volume, futures contract position, hedge rate, the target price and the calculated portfolio price:

```
> head(show(cal06_obpi_b))
        Date Market Trade Exposed Position     Hedge   Target Portfolio
1 2004-01-02  26.82    17      13       17 0.5666667 29.83626  26.82000
2 2004-01-05  26.63    -1      14       16 0.5333333 29.83626  26.73767
3 2004-01-07  26.31     0      14       16 0.5333333 29.83626  26.58833
4 2004-01-08  26.31     0      14       16 0.5333333 29.83626  26.58833
5 2004-01-09  26.54     0      14       16 0.5333333 29.83626  26.69567
6 2004-01-12  26.32     0      14       16 0.5333333 29.83626  26.59300
```

For the sake of comparison, the OBPI strategy for CAL-06 from a sellers point of view can be implemented with similar assumptions by setting the volume to $q = -30$. Using the default at-the-money strike price, the seller calculates an expected target floor to protect at 23.80 EUR/MWh and an initial hedge rate of 43 percent. As the market starts to rise, the hedge is reduced. The seller increases the hedge in late 2004 to dampen the effect from the market drop, and finally exits the forward market positions as the price increases during 2005. The portfolio price follows the market upwards with a premium for the put option replication, as expected for an insurance scheme. The CAL-06 contract closes at 37.81 EUR/MWh, and the seller has a portfolio price of 35.34 EUR/MWh, which may be locked in on the final trading day.

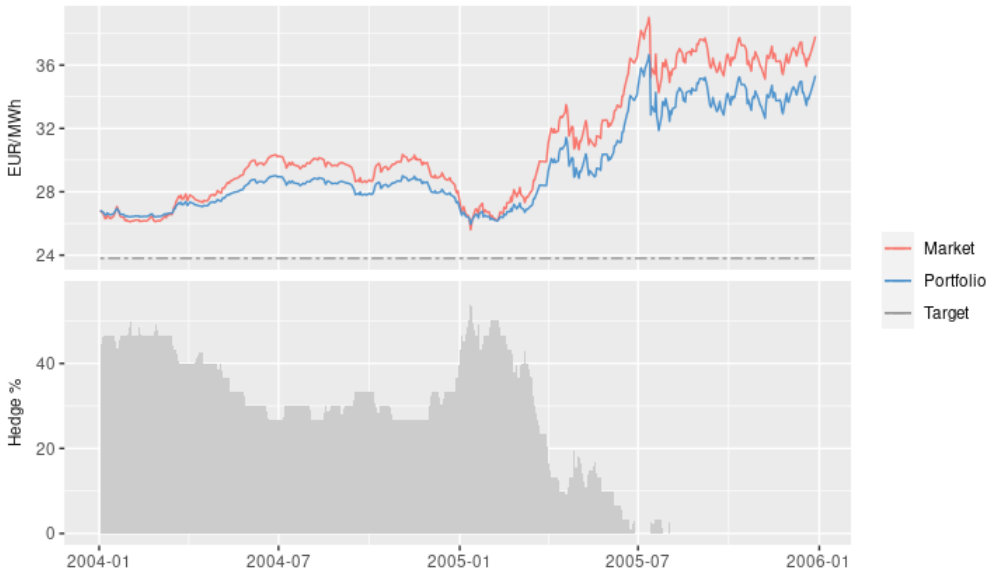


Figure 7: Option based portfolio insurance (OBPI) strategy for seller CAL-06. Daily observations for prices (top panel) and hedge rate (bottom panel). The hedge rate is lowered in the upwards trending market, and the portfolio price continue to increase.

```
# instance of the OBPI class
cal06_obpi_s <- obpi(q = - 30,
                     tdate = dat06$Date,
                     f = dat06$`CAL-06`,
                     k = dat06$`CAL-06`[1],
                     vol = 0.2,
                     r = 0,
                     tdays = 250,
                     daysleft = 500,
                     tcost = 0,
                     int = TRUE)

# the generic plot() method
plot(cal06_obpi_s, title = "", legend = "right", ylab.1 = "EUR/MWh")

> summary(cal06_obpi_s)
$Description
[1] "Hedging strategy of type OBPI and length 500"

$Volume
[1] -30

$Target
[1] 23.80374

$ChurnRate
[1] 4.2

$Stats
      Market Trade Exposed Position     Hedge   Target Portfolio
First  26.82   -13     -17      -13 0.4333333 23.80374  26.82000
Max    39.01     2     -13        0 0.5666667 23.80374  36.64867
Min    25.60   -13     -30      -17 0.0000000 23.80374  25.95167
Last   37.81     0     -30        0 0.0000000 23.80374  35.33567
```

In the examples above, we have implicitly assumed that both the consumer and the seller have a flat volume corresponding to 30 MW over the entire year which can be covered by a base load contract such as the CAL-06. In practice, this is typically not the case. Industrial energy consumers will have consumption profiles determined by the activity level in their production facilities, and often face seasonal shifts due to variation in demand, or holidays. Weather also play a large role, both for consumers and producers such as hydroelectric plants. In order to hedge the predicted volume more precisely, some of the other contract types presented in Table 2 will need to be included in the portfolio. Market players will "roll forward" and start trading contracts covering shorter periods such as quarters, months and weeks, as they become available. The mandate for the energy portfolio will typically be broken down into smaller time intervals with expected volume and required hedge levels. All strategies presented here may be used to make decisions for several years and their sub periods, and the market value of a specific volume prognosis and corresponding futures positions can be evaluated using the forward curve discussed in previous sections.

In order to maintain focus on the strategies themselves, we continue with the baseload example with 30 MW. In Figure 8 we plot results for the remaining four strategies for the consumer hedging with CAL-06. The benchmark strategies `"SHPI"`, and `"SLPI"` follow simple, mechanical patterns. The `"SHPI"` builds a full hedge gradually over the trading period, ending at either the average forward market price for the period, or the target price. This approach will always ensure a full hedge at expiry, without intervention. The `"SLPI"` does not take any positions unless the target is reached, ending either at the target level, or leaving an option to close at the contract expiry price. As the CAL-06 increase significantly during 2005, both end up at the target level.

The `"CPPI"`, and `"DPPI"` strategies are more dynamic and adjust hedge rate according to market developments and the model parameters. As the `"DPPI"` implements a dynamic risk factor, μ_t, the strategies are geared differently. In this example, the `"DPPI"` successfully adjusts the target price downward on one of the first trading days, and achieves a lower portfolio price on last trading day.

A similar overview from a seller's perspective is provided in Figure 9. As the market trends upwards during the hedging period, none of the strategies end up at the initial target price. The `"SHPI"` builds the hedge positions in a step-wise manner, ending up with a portfolio price equal to

Figure 8: Achieved results for the strategies CPPI, DPPI, SHPI and SLPI for buyer CAL-06. Daily observations for prices (top panels) and hedge rate (bottom panels).

the average futures market price for the period. The `"SLPI"` does not enter any positions, leaving an option to lock in market price at expiry. Finally, we can also here see some differences between `"CPPI"`, and `"DPPI"`. This is due to the dissimilar gearing of the portfolios, but also because of the rather frequent adjustments of the target price by `"DPPI"`. In order to protect the higher targets, hedge rate must be increased and `"DPPI"` falls behind `"CPPI"` and ends up at a lower portfolio price for the seller.

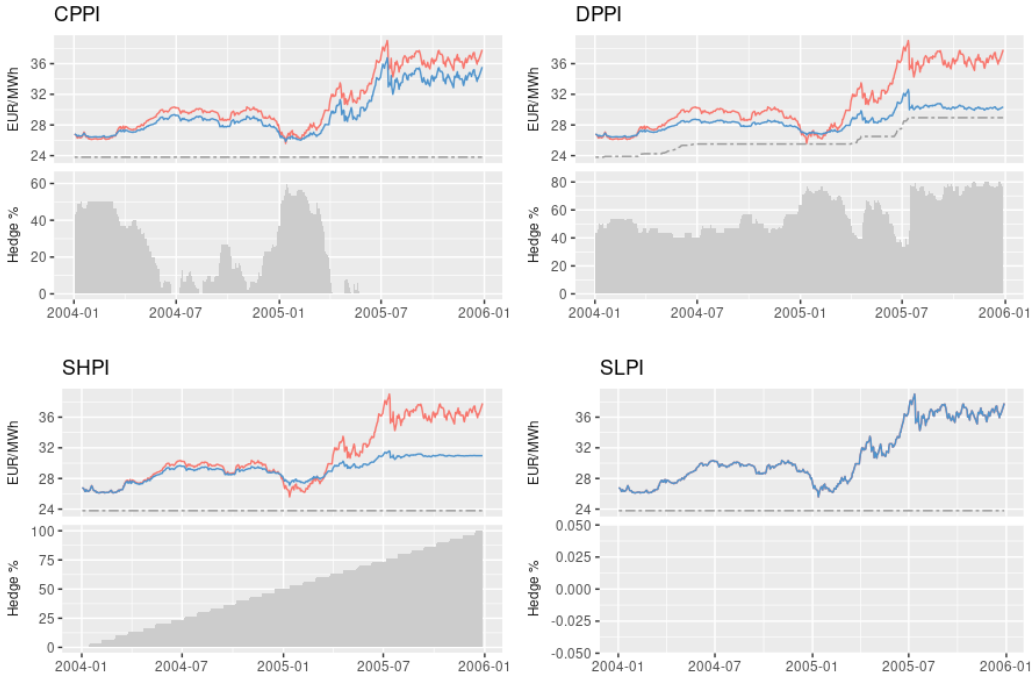


Figure 9: Achieved results for the strategies CPPI, DPPI, SHPI and SLPI for seller CAL-06. Daily observations for prices (top panels) and hedge rate (bottom panels).

etrm can also be used in conjunction with other R packages to evaluate risks related to energy procurement. Metrics such as *Value-at-Risk* and *Expected Shortfall* can for example be calculated using the **PerformanceAnalytics** package. We will proceed with a simple, illustrative example. Consider the OBPI portfolios `"cal06_obpi_b"` and `"cal06_obpi_s"` in the code example above. If we need to calculate risk measures at a specific point in time, say at day 350 in the trading period, we can execute the following code:

```
library(PerformanceAnalytics)

# CAL-06 returns prior to t=350
ret_06 <- head(diff(log(show(cal06_obpi_b)$Market)), 349)

# portfolio status at t=350
pdat <- rbind(
  Buyer =show(cal06_obpi_b)[350,],
  Seller =show(cal06_obpi_s)[350,]
)

# add risk measures to pdat
pdat <-cbind(pdat,
    VaR = abs(rep(VaR(ret_06, p=.95, method="historical"), 2)*pdat$Market*pdat$Exposed*8760),
    ES = abs(rep(ES(ret_06, p=.95, method="historical"), 2)*pdat$Market*pdat$Exposed*8760))
```

The calculation above evaluate market risk related to the unhedged volume (exposed MW $\times$8760 hours in the year 2006) at current market prices under the (simplistic) assumption of symmetry in the returns distribution. The portfolio status, including risk metrics is

```
> pdat[c(-1, -3)]
       Market Exposed Position Hedge   Target Portfolio VaR         ES
Buyer   32.54       3       27   0.9    29.84  28.98     10671.55    18237.06
Seller  32.54     -27       -3   0.1    23.81  30.38    96043.94    164133.52
```

4 Overview of the etrm package

Package **etrm** offers an open source implementation of core functionalities of an ETRM system:

- Construction of forward curves
- Strategies for price risk management

Functions included in the package are listed in Table 5.

Function	Description
`msfc()`	Maximum Smoothness Forward Curve
`cppi()`	Constant Proportion Portfolio Insurance
`dppi()`	Dynamic Proportion Portfolio Insurance
`obpi()`	Option Based Portfolio Insurance
`shpi()`	Step Hedge Portfolio Insurance
`slpi()`	Stop Loss Portfolio Insurance

Table 5: Overview of **etrm** package functions

All functions act as constructors for their corresponding S4 classes, as described in further detail in previous sections. The classes all have generic methods `plot()`, `summary()` and `show()`. Unit tests covering all functions in **etrm** have been implemented using the **testthat** framework introduced in Wickham (2011b).

Three synthetic data sets are included in the package, see Table 6. The `"powfutures130513"` and `"powpriors130513"` data may be used to create forward curves with the `msfc()` function for the trading date 2013-05-13. The portfolio insurance strategies may be tested on the `"powcal"` data set, which contains historical prices for 11 base load power futures.

Data set	Description
`powfutures130513`	Synthetic data for a set of electricity base load futures quoted at 2013-05-13. Closing prices for contracts with weekly, monthly, quarterly and yearly settlement periods
`powpriors130513`	Two simple priors for forward market price curve Daily values for calculation to be used with powfutures130513
`powcal`	Synthetic data set with daily closing prices for 11 electricity base load futures with yearly settlement periods for 2006-2016

Table 6: Overview of **etrm** package data sets

5 Summary and suggestions for future work

This paper introduces **etrm**, an R package for energy market risk management. The package contains tools previously not available in the R ecosystem, such as the `msfc()` function for building a forward curve for energy commodities with flow delivery contracts and strong seasonality. The forward curve is a key decision making tool with many uses, such as pricing non-standard supply agreements, investment decisions and risk management. **etrm** also provides implementations of portfolio insurance strategies for handling price risk, suitable for both long and short hedgers. The functions can be used for back testing strategies on historical futures price data, risk and strategy evaluations, and as decision support tools for trade execution.

The **etrm** package may be developed further by incorporating new elements. First, the forward curve calculation may be done on an hourly level. The bid-ask spread can be used as price constraint for the optimization, as an extension of the current solution based on closing prices. Competing forward curve calculation methods can also be added to the package, and new asset allocation strategies for price risk management could be included.

A further extension of **etrm** functionality can be to implement a `"PORTFOLIO"` class, consisting of a daily volume prognosis covering the full management horizon, supplemented with authorized volumes per (sub)period and hedging strategy objects implemented in accordance with these authorizations. The portfolio object could contain multiple strategy objects for contracts such as "year", "quarter", "month" and "week", depending on the shape of the volume prognosis. This construction can be priced using the forward curve, and portfolio wide risk measures could be calculated via Monte Carlo simulations on the curve.

6 Acknowledgments

We thank the editor and two anonymous reviewers for their constructive comments, which helped us to improve the manuscript. This work has been supported by The Norwegian Research Council via the industrial Ph.D. scheme. The author also wish to thank Montel AS and former colleagues in Kinect Energy Markets AS for interesting discussions and helpful comments.

Bibliography

K. J. Adams and D. R. Van Deventer. Fitting yield curves and forward rate curves with maximum smoothness. *Journal of Fixed Income*, pages 52–62, 1994. URL https://doi.org/10.3905/jfi.1994.408102. [p321]

N. Anderson, F. Breedon, M. Deacon, A. Derry, and G. Murphy. *Estimating and Interpreting the Yield Curve*, volume 6. John Wiley & Sons Incorporated, 1996. [p321]

M. Benini, M. Marracci, P. Pelacchi, and A. Venturini. Day-ahead market price volatility analysis in deregulated electricity markets. In *IEEE Power Engineering Society Summer Meeting*, volume 3, pages 1354–1359 vol.3, 2002. URL https://doi.org/10.1109/PESS.2002.1043596. [p320]

F. E. Benth and M. Schmeck. Pricing and hedging options in energy markets using black-76. *Journal of Energy Markets*, 7(2), 2014. URL https://doi.org/10.21314/JEM.2014.114. [p320]

F. E. Benth, J. Kallsen, and T. Meyer-Brandis. A non-gaussian ornstein–uhlenbeck process for electricity spot price modeling and derivatives pricing. *Applied Mathematical Finance*, 14(2):153–169, 2007a. URL https://doi.org/10.1080/13504860600725031. [p320]

F. E. Benth, S. Koekkebakker, and F. Ollmar. Extracting and applying smooth forward curves from average-based commodity contracts with seasonal variation. *The Journal of Derivatives*, 15(1):52–66, 2007b. URL https://doi.org/10.3905/jod.2007.694791. [p320, 321]

F. E. Benth, J. S. Benth, and S. Koekebakker. *Stochastic Modelling of Electricity and Related Markets*, volume 11. World Scientific, 2008. URL https://doi.org/10.1142/6811. [p320, 323]

H. Bessembinder and M. L. Lemmon. Equilibrium pricing and optimal hedging in electricity forward markets. *The Journal of Finance*, 57(3):1347–1382, 2002. URL https://doi.org/10.1111/1540-6261.00463. [p320]

T. Bjork. *Arbitrage Theory in Continuous Time*. Oxford University Press, 3 edition, 2009. URL https://EconPapers.repec.org/RePEc:oxp:obooks:9780199574742. [p331]

F. Black. The pricing of commodity contracts. *Journal of Financial Economics*, 3(1):167–179, 1976. URL https://doi.org/10.1016/0304-405X(76)90024-6. [p330]

F. Black and R. W. Jones. Simplifying portfolio insurance. *The Journal of Portfolio Management*, 14(1): 48–51, 1987. URL https://doi.org/10.3905/jpm.1987.409131. [p330]

S. Borak and R. Weron. *A Semiparametric Factor Model for Electricity Forward Curve Dynamics*. Humboldt-Universität zu Berlin, Wirtschaftswissenschaftliche Fakultät, 2008. URL http://dx.doi.org/10.18452/4142. [p321]

M. Burger, B. Klar, A. Müller, and G. Schindlmayr. A spot market model for pricing derivatives in electricity markets. *Quantitative Finance*, 4:109–122, 2004. URL https://doi.org/10.1088/1469-7688/4/1/010. [p320]

J.-S. Chen, C.-L. Chang, J.-L. Hou, and Y.-T. Lin. Dynamic proportion portfolio insurance using genetic programming with principal component analysis. *Expert Systems with Applications*, 35(1):273–278, 2008. URL https://doi.org/10.1016/j.eswa.2007.06.030. [p330]

A. Deoras. Electricity load and price forecasting webinar case study, 2021. https://www.mathworks.com/matlabcentral/fileexchange/28684-electricity-load-and-price-forecasting-webinar-case-study [Accessed: 2021-04-06]. [p320]

A. Eydeland and K. Wolyniec. *Energy and Power Risk Management: New Developments in Modeling, Pricing, and Hedging*, volume 97. John Wiley & Sons, 2002. [p320, 321, 329]

S. Farrington. Energy risk software rankings: A different world, 2020. https://www.risk.net/commodities/energy/7511636/energy-risk-software-rankings-a-different-world [Accessed: 2021-04-05]. [p320]

S.-E. Fleten and J. Lemming. Constructing forward price curves in electricity markets. *Energy Economics*, 25(5):409–424, 2003. URL https://doi.org/10.1016/S0140-9883(03)00039-2. [p321]

M. Hildmann, E. Kaffe, Y. He, and G. Andersson. Combined estimation and prediction of the hourly price forward curve. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–8, 2012. URL 10.1109/PESGM.2012.6345333. [p321]

J. Janczura, S. Trück, R. Weron, and R. C. Wolff. Identifying spikes and seasonal components in electricity spot price data: A guide to robust modeling. *Energy Economics*, 38:96–110, 2013. [p320]

D. S. Kirschen and G. Strbac. *Fundamentals of Power System Economics*. John Wiley & Sons, 2018. [p320, 321, 329]

H.-I. Lee, M.-H. Chiang, and H. Hsu. A new choice of dynamic asset management: the variable proportion portfolio insurance. *Applied Economics*, 40(16):2135–2146, 2008. URL https://doi.org/10.1080/00036840600949280. [p330]

H. Leland and M. Rubinstein. The evolution of portfolio insurance. *Dynamic Hedging*, 01 1976. URL https://www.researchgate.net/publication/265430746_The_Evolution_of_Portfolio_Insurance. [p320, 329, 330]

H. E. Leland. Who should buy portfolio insurance? *The Journal of Finance*, 35(2):581–594, 1980. URL http://www.jstor.org/stable/2327419. [p320, 329]

K. G. Lim and Q. Xiao. Computing maximum smoothness forward rate curves. *Statistics and Computing*, 12(3):275–279, 2002. URL https://doi.org/10.1023/A:1020707028156. [p322]

J. H. McCulloch. Measuring the term structure of interest rates. *The Journal of Business*, 44(1):19–31, 1971. URL http://www.jstor.org/stable/2351832. [p321]

M. Nicolosi. Wind power integration and power system flexibility–an empirical analysis of extreme events in germany under the new negative price regime. *Energy Policy*, 38(11):7257–7268, 2010. ISSN 0301-4215. URL https://www.sciencedirect.com/science/article/pii/S0301421510005860. Energy Efficiency Policies and Strategies with regular papers. [p320]

F. Ollmar. An analysis of derivative prices in the Nordic power market. 2003. URL http://hdl.handle.net/11250/164248. [p320, 321, 322, 323]

A. Perold. Constant proportion portfolio insurance. Harvard business school. *Unpublished manuscript*, 1986. [p330]

A. F. Perold and W. F. Sharpe. Dynamic strategies for asset allocation. *Financial Analysts Journal*, 44(1): 16–27, 1988. URL http://www.jstor.org/stable/4479087. [p320, 329]

S. Sundar. Energy trading & risk management with Matlab webinar case study, 2021. https://www.mathworks.com/matlabcentral/fileexchange/28056-energy-trading-risk-management-with-matlab-webinar-case-study [Accessed: 2021-04-06]. [p320]

H. Wickham. ggplot2. *WIREs Computational Statistics*, 3(2):180–185, 2011a. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.147. [p324]

H. Wickham. testthat: Get Started with Testing. *The R Journal*, 3(1):5–10, 2011b. URL https://doi.org/10.32614/RJ-2011-002. [p337]

Anders D. Sleire
Department of Mathematics
University of Bergen
P.O.Box 7803
N-5020 Bergen
Norway
Anders.Sleire@uib.no

fcaR, Formal Concept Analysis with R

by Pablo Cordero, Manuel Enciso, Domingo López-Rodríguez, and Ángel Mora

Abstract Formal concept analysis (FCA) is a solid mathematical framework to manage information based on logic and lattice theory. It defines two explicit representations of the knowledge present in a dataset as concepts and implications. This paper describes an R package called **fcaR** that implements FCA's core notions and techniques. Additionally, it implements the extension of FCA to fuzzy datasets and a simplification logic to develop automated reasoning tools. This package is the first to implement FCA techniques in R. Therefore, emphasis has been put on defining classes and methods that could be reusable and extensible by the community. Furthermore, the package incorporates an interface with the **arules** package, probably the most used package regarding association rules, closely related to FCA. Finally, we show an application of the use of the package to design a recommender system based on logic for diagnosis in neurological pathologies.

1 Introduction

The main goal of knowledge retrieval and knowledge discovery systems is to extract hidden patterns, trends, behaviours, or rules to solve large-impact real-world problems. Usually, these problems present heterogeneous data sources and are at the core of more general decision processes.

A fundamental principle is that the extracted knowledge should provide some understanding of the analyzed data. As computational systems get in critical and sensitive areas such as medicine, the justice system or financial markets, the knowledge becomes much more relevant to make predictions or recommendations or to detect common interest groups or leaders. However, in many cases, the inability of humans to understand the extracted patterns seems problematic.

To represent and retrieve knowledge from datasets, it has become more important to use formal methods based on logic tools. The formal representation of knowledge and the use of logic tools are more suitable for providing understandable answers. Therefore, it can help avoid the lack of interpretability and explainability of the results.

In particular, formal concept analysis (FCA) (Wille, 1982; Ganter and Wille, 1999) is a well-founded mathematical tool, based on lattice theory and logic, which can retrieve and store the knowledge in the form of concepts (analogous to closed itemsets in transactional databases) and implications (association rules with confidence 1). From this perspective, FCA constitutes a framework that complements and extends the study of exact and approximate association rules.

The origins of FCA were devoted to the study of binary datasets (formal contexts) where variables are called attributes. A relevant extension of FCA uses fuzzy sets (Belohlávek and Vychodil, 2016, 2017) to model real-world problems since datasets may contain imprecise, graded or vague information that is not adequately represented as binary values. The fuzzy extension can also model problems with numerical and categorical attributes since these can be scaled to a truth value describing the degree of fulfilment of the attribute.

Some authors have considered the use of FCA in machine learning. Kuznetsov (2004) relates FCA to some mathematical models of machine learning. Ignatov (2017) summarizes the main topics in machine learning and data mining where FCA has been applied: frequent itemset mining and association rules to make classification and clustering. A closer approach appears in Trabelsi et al. (2017), where a method for supervised classification based on FCA is used. The authors extract rules based on concepts generated previously from data. These rules are used to compute the closure of a set of attributes to obtain a classification rule.

From a dataset, FCA can establish maximal clusters, named concepts, between objects and attributes. Each cluster consists of objects having common properties (attributes), which are only fulfilled for these objects. The hierarchy between the concepts and relationships between the attributes (rules or implications) are computed with the same computational cost in FCA.

Among all the techniques used in other areas to extract knowledge, we emphasize using rules for its theoretical and practical interest. The notion of if-then rules, with different names, appears in several areas (databases, machine learning, data mining, formal concept analysis) as a relationship between attributes, called items, properties or atomic symbols regarding the domain. Nowadays, the number of rules extracted even from medium-sized datasets is enormous in all these areas. Therefore, the intelligent manipulation of these rules to reason with them is a hot topic to be explored.

In this direction, Cordero et al. (2002) introduced a logic, named simplification logic for functional dependencies (SL_{FD}), firmly based on a simplification rule, which allows us to narrow the functional dependency set by removing redundant attributes. Although the semantic of implications or if-then

	P1	P2	P3	P4
O1	0	½	½	½
O2	½	1	0	1
O3	½	1	0	1
O4	0	½	1	½

Table 1: A sample formal context. The attributes are P1 to P4 and the objects are named O1 to O4.

rules in other areas are different, the logic can be used too.

Using directly SL_{FD}, some automated deduction methods directly based on this inference system have been developed for classical systems and fuzzy systems (Mora et al., 2003; Cordero et al., 2012; Mora et al., 2012; Rodríguez Lorenzo et al., 2014, 2015).

In the fuzzy framework, several approaches to the definition of fuzzy implications (functional dependencies, rules) are proposed in the literature, see Jezková et al. (2017) for a survey. Our work considers that the definition of graded implication proposed by Belohlávek and Vychodil (2016, 2017) generalizes all the previous definitions. Furthermore, for this general definition of graded implications, an axiomatic system named FASL (fuzzy attribute simplification logic) was developed by Belohlávek et al. (2016), becoming a helpful reasoning tool. Note that FASL is a generalization of SL_{FD} to the fuzzy framework.

The core of our proposal is to provide a user-friendly computational interface to the principal operators and methods of fuzzy FCA, including the mentioned logic tools. This interface is easy to extend to new functionalities and incorporate new methods, such as minimal generators or the computation of different implication bases quickly. The operators and methods implemented are designed to work in the general fuzzy setting, but they are also applicable in the classical binary case.

Thus, the focus of our proposal, the **fcaR** package, is to provide easy access to formal methods to extract all the implicit knowledge in a dataset in the form of concepts and implications, working natively with fuzzy sets and fuzzy implications. Our goal is to provide a unified computational framework for the theoretically-oriented FCA users to develop, test, and compare new methods and knowledge extraction strategies.

Other objectives of this work include presenting an FCA-based tool for knowledge discovery accessible to other scientific communities, allowing for the development of new packages in other fields using FCA techniques, especially in the construction of recommendation systems.

The work is organized as follows: Section Background on FCA begins with a brief look of FCA and Simplification Logic. Section Related works presents other software libraries that implement FCA or related paradigms' core notions. In Section Package design, an explanation of the data structures, classes and constructor methods is covered. Section Formal concept analysis with fcaR shows how to use the package, describing the implemented FCA methods and the use of the simplification logic. In Section Usage example. Fuzzy diagnostic system, a real application of the package in developing a recommender system is illustrated. Finally, some conclusions and future works are presented in Section Conclusions and future work.

2 Background on FCA

In this section, we present the basic notions in the FCA framework using a running example (see Table 1). Note that the package incorporates the main methods in FCA that appear in this summary. Since the formal study of FCA is not the main scope of this work, we recommend the reference (Ganter and Obiedkov, 2016) for further details of this framework.

A *formal context* is a triple (G, M, I), where G is a set of objects, M is a set of attributes and I is a fuzzy relation between objects and attributes, where $I(x, y) \in [0, 1]$ means the truth value to which object x possesses attribute y, indicating $I(x, y) = 0$ the absence of attribute or property y in object x.

The meaning of each entry in the table is the extent to which an object possesses the attribute in the corresponding column. In the example shown in Table 1, the object named O4 fully possesses attribute P3 and possesses P2 and P4 only to degree 50%.

In the remaining of this paper, we will use the notation $^d/a$ to indicate the presence of attribute a with degree d.

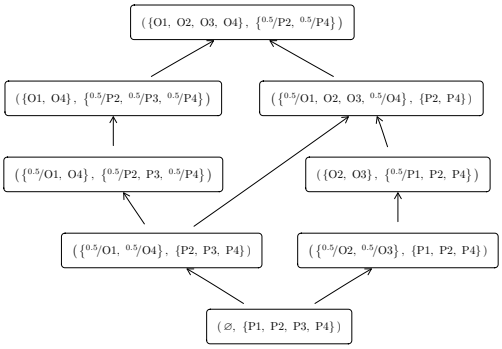


Figure 1: Concept lattice for the context in Table 1. Arrows indicate the direction of the order relationship between concepts.

Derivation operators

Given a fuzzy set of objects $\mathcal{S}$, we can compute its *intent* as the set of attributes that are shared by all objects in $\mathcal{S}$. Analogously, we define the *extent* of a set $\mathcal{T}$ of attributes as the set of objects which have *all* the attributes in $\mathcal{T}$.

In the above example, for $S = \{\text{O1, O2}\}$, we have $\text{intent}(S) = \{^{0.5}/\text{P2, } ^{0.5}/\text{P4}\}$ because $I(\text{O1, P2}) = 0.5$, $I(\text{O1, P4}) = 0.5$, $I(\text{O2, P2}) = 1$ and $I(\text{O2, P4}) = 1$.

The operator ϕ defined by $\phi(T) = \text{intent}(\text{extent}(T))$ is a closure operator and a set of attributes $\mathcal{T}$ is called *closed* if $\mathcal{T} = \phi(\mathcal{T})$. In our example, $\phi(\{^{0.5}/\text{P1}\}) = \{^{0.5}/\text{P1, P2, P4}\}$, meaning that *every object* that has P1 with degree at least 0.5, also has all the attributes {P2, P4}. When $\mathcal{T}$ is closed, the pair $(\text{extent}(\mathcal{T}), \mathcal{T})$ is called a *concept*.

In general, a concept (A, B), where A is a set of objects, and B is a set of attributes, means that the only attributes shared by all objects in A are those in B, and the only objects having all attributes in B are those in A. This property makes (A, B) a *maximal rectangular cluster* in the dataset, with a strong dependence between the objects in A and the attributes in B. In the formal context represented in Table 1, the pair $(\{\text{O2, O3}\}, \{^{0.5}/\text{P1, P2, P4}\})$ is a concept, because $\text{extent}(\{^{0.5}/\text{P1, P2, P4}\}) = \{\text{O2, O3}\}$ and $\text{intent}(\{\text{O2, O3}\}) = \{^{0.5}/\text{P1, P2, P4}\}$, i.e. a fixpoint is achieved.

In FCA, using these derivation operators, two operations can be used to reduce a formal context:

- *Clarification*, which is the removal of duplicated rows (objects) and columns (attributes) of the formal context since duplicates do not contribute knowledge to the context.
- *Reduction*, which removes attributes that can be expressed as the closure of other attributes.

These two operations remove redundancies in the formal context without affecting the knowledge contained in it. Many of the subsequent operations in FCA have high computational complexity, and clarifying and reducing a formal context may reduce the computational time of posterior operations.

The concept lattice

The *concept lattice* of a formal context is the set of all concepts, with the partial order $\leq$ defined as follows: for two concepts (A_1, B_1) and (A_2, B_2), we say that $(A_1, B_1) \leq (A_2, B_2)$ if and only if $A_1 \subseteq A_2 \iff B_2 \subseteq B_1$. For instance, $(\{^{0.5}/\text{O2, } ^{0.5}/\text{O3}\}, \{\text{P1, P2, P4}\}) \leq (\{\text{O2, O3}\}, \{^{0.5}/\text{P1, P2, P4}\})$, since their intents verify $\{^{0.5}/\text{P1, P2, P4}\} \subseteq \{\text{P1, P2, P4}\}$.

This order (or precedence) relationship induces a hierarchy of concepts, which can be graphically represented in the Hasse diagram (Birkhoff, 1940) of the partially ordered set of concepts. In Figure 1, we find the Hasse diagram corresponding to our running example.

From the concept lattice and the order relationship, we can define notions as *subconcept*, *superconcept*, *infimum* and *supremum* of a set of concepts.

Notably, there are the *irreducible elements*, for the infimum (meet) or the supremum (join) operators, which, in the lattice, are shown as those elements with only one arrow departing or arriving at them, respectively. These elements are essential since one can reconstruct the whole lattice by operating with these elements.

The *standard context* for a given formal context $\mathbb{K}$ is another formal context $(\mathcal{J}, \mathcal{M}, \leq)$, where $\mathcal{J}$ and $\mathcal{M}$ are the sets of join- and meet-irreducible elements of $\mathbb{K}$ and $\leq$ is the partial order defined in the concept lattice. This standard context has a concept lattice isomorphic to that of $\mathbb{K}$.

Implications and logic

The knowledge stored in a formal context can also be represented as a set of implications, which are expressions of the form $A \Rightarrow B$ where A and B are sets of attributes or items, indicating that, for every object in which the set of attributes A is present, also B is present. This interpretation is similar to the one defined in data mining/machine learning over the so-named association rules. The confidence (a well-known estimator of the rules' quality) has value 1 in all the implications.

For instance $\{^{0.5}/\text{P1}\} \Rightarrow \{\text{P4}\}$ is a valid implication in the previous example, having the following interpretation: when the attribute P1 has degree at least 0.5 then we have P4 with degree 1.

The *Duquenne-Guigues basis* of implications (Guigues and Duquenne, 1986) is a set of valid implications from which all other valid implications can be deduced. The Duquenne-Guigues basis in our example is given by:

$$\begin{array}{lrcl} 1: & \varnothing & \Rightarrow & \{^{0.5}/\text{P2}, {}^{0.5}/\text{P4}\} \\ 2: & \{^{0.5}/\text{P2}, \text{P4}\} & \Rightarrow & \{\text{P2}\} \\ 3: & \{\text{P2}, {}^{0.5}/\text{P4}\} & \Rightarrow & \{\text{P4}\} \\ 4: & \{\text{P2}, {}^{0.5}/\text{P3}, \text{P4}\} & \Rightarrow & \{\text{P3}\} \\ 5: & \{^{0.5}/\text{P1}, {}^{0.5}/\text{P2}, {}^{0.5}/\text{P4}\} & \Rightarrow & \{\text{P2}, \text{P4}\} \\ 6: & \{^{0.5}/\text{P1}, \text{P2}, \text{P3}, \text{P4}\} & \Rightarrow & \{\text{P1}\} \end{array}$$

In Cordero et al. (2002), the simplification logic, denoted as SL_{FD}, was introduced as a method to manipulate implications (functional dependencies or if-then rules), removing redundancies or computing closures of attributes. This logic is equivalent to Armstrong's Axioms (Armstrong, 1974), which are well known from the 80s in databases, artificial intelligence, formal concept analysis, and others. The axiomatic system of this logic considers reflexivity as the axiom scheme

$$\texttt{[Ref]} \quad \frac{A \supseteq B}{A \Rightarrow B}$$

together with the following inference rules called fragmentation, composition and simplification, respectively, which are equivalent to the classical Armstrong's axioms of augmentation and, more importantly, transitivity.

$$\texttt{[Frag]}\ \frac{A \Rightarrow B \cup C}{A \Rightarrow B} \qquad \texttt{[Comp]}\ \frac{A \Rightarrow B,\ C \Rightarrow D}{A \cup C \Rightarrow B \cup D} \qquad \texttt{[Simp]}\ \frac{A \Rightarrow B,\ C \Rightarrow D}{A \cup (C \setminus B) \Rightarrow (D \setminus B)}$$

The main advantage of SL_{FD} with respect to Armstrong's Axioms is that the inference rules may be considered as equivalence rules, (see the work by Mora et al. (2012) for further details and proofs), that is, given a set of implications Σ, the application of the equivalences transforms it into an equivalent set. In the package presented in this paper, we develop the following equivalences:

1. Fragmentation Equivalency **[FrEq]**: $\{A \Rightarrow B\} \equiv \{A \Rightarrow B \setminus A\}$.
2. Composition Equivalency **[CoEq]**: $\{A \Rightarrow B, A \Rightarrow C\} \equiv \{A \Rightarrow B \cup C\}$.
3. Simplification Equivalency **[SiEq]**: If $A \subseteq C$, then
$$\{A \Rightarrow B, C \Rightarrow D\} \equiv \{A \Rightarrow B, A \cup (C \setminus B) \Rightarrow D \setminus B\}$$
4. Right-Simplification Equivalency **[rSiEq]**: If $A \subseteq D$, then
$$\{A \Rightarrow B, C \Rightarrow B \cup D\} \equiv \{A \Rightarrow B, C \Rightarrow D\}$$

Usually, many areas, the implications have always atomic attributes on the right-hand side. We emphasize that this logic can manage *aggregated* implications, i.e. the implications' consequents do not have to be singletons. This represents an increase of the logic efficiency.

This logic removes attribute redundancies in some of the implications in the Duquenne-Guigues basis presented before. Particularly, the implications with numbers 2, 3, 4, 5 and 6 are simplified to:

$$\begin{array}{lrcl} 2: & \{\text{P4}\} & \Rightarrow & \{\text{P2}\} \\ 3: & \{\text{P2}\} & \Rightarrow & \{\text{P4}\} \\ 4: & \{^{0.5}/\text{P3}, \text{P4}\} & \Rightarrow & \{\text{P3}\} \\ 5: & \{^{0.5}/\text{P1}\} & \Rightarrow & \{\text{P4}\} \\ 6: & \{^{0.5}/\text{P1}, \text{P3}\} & \Rightarrow & \{\text{P1}\} \end{array}$$

One of the primary uses of a set of implications is computing the closure of a set of attributes, the maximal fuzzy set that we can arrive at from these attributes using the given implications.

The importance of computing the closure from implications is because implications can be managed using logic tools. They formally describe the knowledge existing in the dataset; thus, the user can forget about the original *dataset*. In this sense, it is similar to supervised Machine Learning techniques.

The algorithm to compute the closure (Mora et al., 2012) is based on the classical CLOSURE algorithm (Maier, 1983; Ganter and Obiedkov, 2016). After each pass over the set of implications, instead of simply removing the implications used in the step, the simplification rule substitutes the implications by others equivalent but simpler, with fewer attributes. In the binary case, the reduced set of implications does not reference any attribute in the computed closure. A detailed description of the procedure is presented in Algorithm 1.

Algorithm 1: SL_{FD} closure

Input : X: attribute set; Γ: set of implications
Output: X^+: the closure of X with respect to Γ; Γ': the simplified set of implications

$\Gamma' := \Gamma \cup \{\varnothing \Rightarrow X\}$
$X_{\text{new}} := X$
$X_{\text{old}} := X$
repeat
  Replace $\{\varnothing \Rightarrow X_{\text{old}}\}$ with $\{\varnothing \Rightarrow X_{\text{new}}\}$ in Γ'
  $X_{\text{old}} = X_{\text{new}}$
  for *each* $A \Rightarrow B \in \Gamma' \setminus \{\varnothing \Rightarrow X_{\text{new}}\}$ **do**
    if $A \subseteq X_{\text{new}}$ **then**
      Replace $\{\varnothing \Rightarrow X_{\text{new}}\}$ with $\{\varnothing \Rightarrow X_{\text{new}} \cup B\}$
      $X_{\text{new}} := X_{\text{new}} \cup B$
    end
    if $B \subseteq X_{\text{new}}$ **then**
      Remove $A \Rightarrow B$ from Γ'
    end
    if $A \cap X_{\text{new}} \neq \varnothing$ *or* $B \cap X_{\text{new}} \neq \varnothing$ **then**
      Replace $A \Rightarrow B$ with $A \setminus X_{\text{new}} \Rightarrow B \setminus X_{\text{new}}$
    end
  end
until $X_{\text{old}} = X_{\text{new}}$
return X^+ *and* Γ'

For instance, the closure of $S = \{^{0.5}/\text{P2}\}$ with this algorithm is $S^+ = \{^{0.5}/\text{P2}, {}^{0.5}/\text{P4}\}$ (this means that all objects that have all the attributes in S, also have those in S^+). The simplification logic leads us to a reduced set of implications Γ':

1:	{P4}	⇒	{P2}
2:	{P2}	⇒	{P4}
3:	{P2, $^{0.5}$/P3, P4}	⇒	{P3}
4:	{$^{0.5}$/P1}	⇒	{P2, P4}
5:	{$^{0.5}$/P1, P2, P3, P4}	⇒	{P1}

One can interpret these implications as the knowledge in the original implications if the attributes in S are not considered (formally, this is equivalent to suppose that $\varnothing \Rightarrow S$ is true). In the example, if, in addition to having $\{^{0.5}/\text{P2}\}$, we have {P1} with degree 0.5, we can infer that {P2} and {P4} are fully present, by implication number 4.

3 Related works

The package presented in this paper has been developed in the R language. In recent years, R has lived a *remarkable revolution* caused in part by an increasing user community from many different scientific fields. This has led to the development of reproducible research tools, e.g., **rmarkdown** (Xie et al., 2018), and tools to connect and interact with other programming languages, with packages like **Rcpp** (Eddelbuettel and François, 2011) or **reticulate** (Ushey et al., 2020). Besides, the development of a game-changer programming paradigm with *tidy* principles has provided a significant impact on R's usability, see the **tidyverse** of Wickham et al. (2019).

These facts have transformed R from a purely statistical language into one of the most popular languages in data science, machine learning, data mining or visualization. In R, there are multiple packages to perform data mining and machine learning tasks. However, only a few of them are

	ToscanaJ	ConExp	conexp-clj	Galicia	GALACTIC	**arules**	**RKEEL**	**frbs**	**fcaR**
Programming language	Java	Java	Clojure	Java	Python	R	R	R	R
Context operations		×	×	×	×				×
Context visualization		×	×	×	×				×
Lattice computation		×	×	×	×				×
Lattice operations	×	×	×	×	×				×
Lattice visualization	×	×	×	×	×				×
Implication basis computation		×	×		×				×
Association rules computation		×	×	×	×	×	×	(1)	(2)
Closure operators			×		×				×
Use of logic tools									×
Native fuzzy sets			(3)		(3)				×

Table 2: Functionality comparison among different software libraries for FCA. (1) The rules can be used only for classification or regression; (2) **fcaR** is integrated with **arules** to import and export exact association rules; (3) some packages do not use natively fuzzy sets as representation, but use scaled or discretized contexts.

focused on formal methods that can model and extract knowledge in the form of implications and rules on fuzzy sets and operate on them using logic tools.

The **arules** package by Hahsler et al. (2005) provides the infrastructure for representing, manipulating and analyzing transaction data and patterns (frequent itemsets and association rules) in *binary* settings. The **frbs** package (Riza et al., 2015) presents an implementation of various learning algorithms based on fuzzy rule-based systems (FRBSs) for dealing with classification and regression tasks. The **RKEEL** package (Moyano and Sanchez Ramos, 2016) is an R interface to KEEL, a popular Java software for a large number of different knowledge data discovery tasks. It can extract association rules from numerical datasets, but variables are discretized or categorized first.

It must be noted that none of the previous approaches uses logic tools to infer knowledge from the extracted rules and implications. Also, the support for fuzzy sets is minimal, and none of them implements the core notions of FCA.

Since the **arules** package has become a *de facto* standard, one of the critical points in our proposal's design has been to get **fcaR** easily integrated with **arules**.

In other programming languages, several libraries are implementing FCA methods. Some of the most used libraries are focused on computing the concept lattice and provide a graphical interface to operate with the formal context and the concept lattice: ToscanaJ (Becker and Correia, 2005) and Galicia (Valtchev et al., 2003). Other tools, such as Concept Explorer (ConExp) (Yevtushenko, 2000), with its new versions ConExp NG and ConExp FX, besides their graphical capabilities, implement the core FCA algorithms and operations: context editing, building concept lattices, finding bases of implications and association rules, for instance.

The two most recent and fully-featured libraries are conexp-clj (Hanika and Hirth, 2019) and GALACTIC (Demko and Bertet, 2020), focusing on their extensibility and addition of multiple new algorithms, including the implementation of closure operators and methods based on implications.

In Table 2, we present a comparison of the main features present in each of these libraries.

In conclusion, **fcaR** can be considered among the general-purpose FCA tools with a more comprehensive feature list, integrating the core notions of FCA and logic tools.

4 Package design

In this section, we present the design principles and implementation information about the **fcaR** package: the use of object-oriented programming, the need to be integrated with the **arules** package, and its use in reproducible research.

Package availability

This package is available in CRAN and can be installed using `install.packages('fcaR')`. Also, the development version with new features and bugfixes can be installed from GitHub using `remotes::install_github('Malaga-FCA-group/fcaR')`.

All the documentation in the form of a **pkgdown** site can be found in https://malaga-fca-group.github.io/fcaR/.

Class name	Use
`"Set"`	A basic class to store a fuzzy set using sparse matrices
`"Concept"`	A pair of sets (extent, intent) forming a concept for a given formal context
`"ConceptLattice"`	A set of concepts with their hierarchical relationship. It provides methods to compute notable elements, sublattices and plot the lattice graph
`"ImplicationSet"`	A set of implications, with functions to apply logic and compute closure of attribute sets
`"FormalContext"`	It stores a formal context, given by a table, and provides functions to use derivation operators, simplify the context, compute the concept lattice and the Duquenne-Guigues basis of implications

Table 3: Main classes found in **fcaR**.

Data structures

Now, we present an overview of the data structures implemented in **fcaR** and the design principles on which the development of the package has been based. The main points are the **R6** (Chang, 2020) object-oriented programming (OOP) paradigm and the extensive use of sparse matrices for internal storage of objects.

The **R6** OOP paradigm is being increasingly used in the R ecosystem since its ability to encapsulate data structures and methods related to each class in a single and straightforward interface exposed to the user. This reason and its extensibility have made it the choice to implement the data structures in this package.

The core of the **fcaR** package provides data structures that allow the user to work seamlessly with formal contexts and sets of concepts and implications. The main classes are presented in Table 3. Let us briefly describe the internal representation of the main object classes.

Formal contexts

A formal context (G, M, I) can be represented by a two-way table I indicating the relationship between objects and attributes. This table is expressed in matrix form in R, with rows corresponding to objects and columns corresponding to attributes.

In the classical setting, as occurred in the **arules** package, a formal context may represent a collection of *itemsets* which conform to a *transactions database*. In such a setting, the matrix is binary, and each of its entries represents the presence (1) or absence (0) of an item in a particular itemset. A value of 1 in the matrix is interpreted as the object fully possessing the associated attribute.

In this package, one of the main features is dealing with fuzzy or graded attributes. Therefore I is not a binary matrix anymore, but a numerical matrix with entries in the interval $[0, 1]$. We emphasize that our package can deal with binary datasets as a particular case of the fuzzy case. Most of the methods implemented work for binary and fuzzy Formal Concept Analysis.

The **R6** class `"FormalContext"` stores the relationship matrix (or *incidence* matrix, if binary) I in a compressed sparse format, given by class `"dgCMatrix"` from package **Matrix** (Bates and Maechler, 2021). The reason to store I as a sparse matrix is that, in practice, most situations will occur with many objects, and only a few attributes present for each object. This allows for greater computational efficiency.

The main methods applicable to an object of the `"FormalContext"` class are presented in Table 4.

The features of the **R6** paradigm allow us to build another critical aspect of a `"FormalContext"`: it stores both the associated `"ConceptLattice"` and the Duquenne-Guigues basis as an `"ImplicationSet"`, which can be accessed using `fc$concepts` and `fc$implications`, respectively. Initially, these instances are empty (no concepts nor implications are included), and they are filled with the corresponding data as needed.

Concept lattices

A `"ConceptLattice"` object is usually built from a `"FormalContext"` or by using some specific methods on an existing `"ConceptLattice"`. Thus, although it has a constructor method, the user will not likely use it to create an instance of this class.

Internally, the `"ConceptLattice"` class stores three sparse matrices: two for the extents and intents in matrix form (rows are objects/attributes and columns represent concepts) and a sparse matrix indicating the order relationship between concepts. The element (i, j) of this matrix is 1 if and only if

Method name	Purpose
`new(I)`	Constructor for the `"FormalContext"` class. `I` can be a `"matrix"`, a `"data.frame"` or a filename from where to import the context
`intent(S)`	Computes the intent of a set `S` of objects
`extent(S)`	Computes the extent of a set `S` of attributes
`closure(S)`	Computes the closure of a set `S` of attributes
`clarify()`	Performs clarification of a `"FormalContext"`
`reduce()`	Performs reduction of a `"FormalContext"`
`standardize()`	Builds the standard context $(\mathcal{J}, \mathcal{M}, \leq)$ for the given `"FormalContext"`
`find_concepts()`	Builds the concept lattice following the NextClosure algorithm for fuzzy formal contexts
`find_implications()`	Uses the modified NextClosure for fuzzy formal contexts to compute the concept lattice and the Duquenne-Guigues basis of implications
`to_transactions()`	Converts the formal context to an object of class `"transactions"` from the **arules** package

Table 4: Main methods of the `"FormalContext"` class.

the *i*-th concept is a subconcept of the *j*-th concept, otherwise it is 0.

The main methods of objects of the `"ConceptLattice"` class are summarized in Table 5.

Method name	Purpose
`new(A,B)`	Constructor for the `"ConceptLattice"` class. `A` and `B` are the extents and intents of the concepts
`intents()`	Retrieves the intents of all the concepts
`extents()`	Retrieves the extents of all the concepts
`sublattice(C)`	Computes the smallest sublattice that includes all concepts in the concept set `C`
`meet_irreducibles()`, `join_irreducibles()`	Compute the meet-irreducible and join-irreducible elements of the lattice
`decompose(C)`	Decomposes the concept `C` as the supremum of meet-irreducible concepts
`supremum(C)`, `infimum(C)`	Compute the supremum or infimum of a set `C` of concepts
`subconcepts(C)`, `superconcepts(C)`	Compute the subconcepts and superconcepts of a concept `C`
`lower_neighbours(C)`, `upper_neighbours(C)`	For a given concept `C`, compute its lower and upper neighbours in the lattice

Table 5: Main methods of the `"ConceptLattice"` class.

Implication sets

There are two ways in which an `"ImplicationSet"` can be created: by finding the implications of a given `"FormalContext"` or by importing a `"rules"` object from the **arules** package.

In both cases, internally, an `"ImplicationSet"` object is composed of two sparse matrices, `lhs_matrix` and `rhs_matrix`, representing the left-hand and right-hand sides (LHS and RHS, respectively) of the computed implications. As it is the default in **Matrix**, these matrices are stored column-wise, such that column *j* represents the *j*-th implication in the set, and row *i* stands for the *i*-th attribute.

The main methods applicable to an object of class `"ImplicationSet"` are described in Table 6.

Method name	Purpose
`new(A,B)`	Constructor of the `"ImplicationSet"` class. `A` and `B` represent the left-hand and right-hand sides of the implications
`add(P)`	Concatenates the implications in the current set with those in `P`
`closure(S)`	Computes the closure of the attribute set `S` with respect to the current implication set, applying the SL_{FD} logic
`recommend(S,attr)`	Computes the closure of `S` and filters it to show only the attributes in `attr`
`apply_rules(rules)`	Transforms the `"ImplicationSet"` into another using equivalence rules
`to_basis()`	Transforms the `"ImplicationSet"` to an equivalent basis of implications
`filter(lhs,rhs)`	Filters the `"ImplicationSet"` to retrieve only implications with specific attributes in the left-hand or in the right-hand sides
`to_arules()`	Converts a binary `"ImplicationSet"` to the `"rules"` class of **arules**

Table 6: Main methods of the `"ImplicationSet"` class.

Interfacing with arules

One of the main design objectives in this package is interoperability with the **arules** package since it can be considered a standard in the field.

The constructor methods for the `"FormalContext"` and `"ImplicationSet"` classes allow as inputs objects of types `"transactions` and `"rules"`, respectively, from the **arules** package.

A `"transactions"` object in **arules** can be represented by a binary formal context. Each transaction can be translated into one object as defined in FCA. In contrast, different transaction items are mapped to binary variables according to whether the item is present or not in the transaction. Any object of type `"transactions"` can be used to initialize a `"FormalContext"`, assigning the labels of the items present in the *transaction database* to the attributes' names in the context, and using the `"itemMatrix"` as the internal representation of the binary incidence matrix. Also, **fcaR** can export the `"FormalContext"` to the `"transactions"` format, by using the `to_transactions()` method of a binary `"FormalContext"`.

Analogously, a `"rules"` object can be used to create a `"ImplicationSet"`. The LHS and RHS of the `"rules"` object are mapped to the `lhs_matrix` and `rhs_matrix` of the `"ImplicationSet"`. In addition, the method `to_arules()` can be used to convert any binary `"ImplicationSet"` to the `"rules"` format.

These interfaces allow the users who usually work in other areas with **arules** to use FCA methods and algorithms for transactions databases, as an extension of what can do in **arules**, and then, if necessary, to convert back to the **arules** format.

Use for reproducible research

Another of the design goals of the **fcaR** package is that it could be used in research and used in publications. All the implemented classes in **fcaR** provide a `to_latex()` function so that concepts, implications and the table of the formal context can be easily exported to LaTeX. Also, the `"FormalContext"` and `"ConceptLattice"` classes have a `plot()` method that allows to graphically represent that object types. This `plot()` function allows to export the graphs in LaTeX format to be included directly in PDF reports with publication quality (see the Figures in this work).

All this document has been written using the Rmarkdown format using these functions to generate the tables and listings of concepts and implications. The only required LaTeX packages are `amssymb` (for some mathematical symbols), `longtable` and `caption` (for the listings of concepts and implications) and `tikz` (for plotting formal contexts and concept lattices).

5 Formal concept analysis with fcaR

In this Section, we will show how to perform the basic operations mentioned in Section Background on FCA. We will use the same formal context shown in Table 1.

Let us suppose that the variable 'I' stores the matrix corresponding to that formal context. To create a new `"FormalContext"` with that matrix, the command is `fc <- FormalContext$new(I)`. Now, the object `fc` stores the formal context and enables us to perform the corresponding operations on it.

For example, we can access the attribute and object names with `fc$attributes` and `fc$objects`, respectively. We can even plot a heatmap (see Fig. 2) to show the sparsity of the context, very useful for large contexts.

Derivation operators

The methods that implement the derivation operators are named after them: `intent()`, `extent()` and `closure()`. They can be applied on objects of type `"Set"`, representing fuzzy sets of objects or attributes:

```
> S <- Set$new(fc$objects, O1 = 1, O2 = 1)
> S
{O1, O2}
> fc$intent(S)
{P2 [0.5], P4 [0.5]}
> T <- Set$new(fc$attributes, P1 = 1, P3 = 1)
> T
{P1, P3}
> fc$extent(T)
{}
```

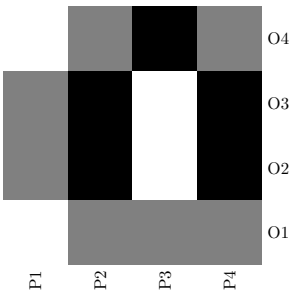


Figure 2: The heatmap representing the formal context shown in Table 1, obtained by doing `fc$plot()`. A grey scale is used, where white indicates the value 0 for an object-attribute pair, while black indicates the value 1, with the greys representing the intermediate values. This type of graph aids visual inspection of the context, being able to check its sparsity, density or possible patterns.

```
> fc$closure(T)
{P1, P2, P3, P4}
```

In addition, we can perform *clarification* on the formal context, by using `fc$clarify()`, giving:

```
FormalContext with 3 objects and 3 attributes.
           P1   P3  [P2, P4]
       O1  0   0.5     0.5
       O4  0    1      0.5
 [O2, O3] 0.5   0       1
```

The duplicated rows and columns in the formal context have been collapsed, and the corresponding attributes and objects' names are grouped together between brackets, e.g., [P2, P4].

Concept lattice

The command to compute the concept lattice for a `"FormalContext"` `fc` is `fc$find_concepts()`. The lattice is stored in `fc$concepts`, which is of the `"ConceptLattice"` class.

```
> fc$concepts
A set of 8 concepts:
1: ({O1, O2, O3, O4}, {P2 [0.5], P4 [0.5]})
2: ({O1, O4}, {P2 [0.5], P3 [0.5], P4 [0.5]})
3: ({O1 [0.5], O4}, {P2 [0.5], P3, P4 [0.5]})
4: ({O1 [0.5], O2, O3, O4 [0.5]}, {P2, P4})
5: ({O1 [0.5], O4 [0.5]}, {P2, P3, P4})
6: ({O2, O3}, {P1 [0.5], P2, P4})
7: ({O2 [0.5], O3 [0.5]}, {P1, P2, P4})
8: ({}, {P1, P2, P3, P4})
```

In order to know the *cardinality* of the set of concepts (that is, the number of concepts), we can use `fc$concepts$size()`, which gives 8 in this case. The complete list of concepts can be printed with `fc$concepts$print()`, or simply `fc$concepts`. Also, they can be translated to LaTeX using the `to_latex()` method, as mentioned before.

The typical subsetting operation in R with brackets is implemented to select specific concepts from the lattice, giving their indexes or a boolean vector indicating which concepts to keep. The same rules for subsetting as in R base apply:

```
> fc$concepts[c(1:3, 5, 8)]
A set of 5 concepts:
1: ({O1, O2, O3, O4}, {P2 [0.5], P4 [0.5]})
2: ({O1, O4}, {P2 [0.5], P3 [0.5], P4 [0.5]})
3: ({O1 [0.5], O4}, {P2 [0.5], P3, P4 [0.5]})
4: ({O1 [0.5], O4 [0.5]}, {P2, P3, P4})
5: ({}, {P1, P2, P3, P4})
```

In addition, the user can compute concepts' *support* (the proportion of objects whose set of attributes contains the intent of a given concept) by means of `fc$concepts$support()`.

```
> fc$concepts$support()
[1] 1.00 0.50 0.25 0.50 0.00 0.50 0.00 0.00
```

Sublattices

When the concept lattice is too large, it can be useful to work with a sublattice of the complete lattice on certain occasions. To this end, we use the `sublattice()` function.

For instance, to build the sublattice generated by the concepts $\left(\left\{^{0.5}/\text{O1}, {}^{0.5}/\text{O4}\right\}, \{\text{P2, P3, P4}\}\right)$ and $\left(\{\text{O2, O3}\}, \left\{^{0.5}/\text{P1, P2, P4}\right\}\right)$, which had indexes 5 and 6 in the list of concepts, we can do:

```
> fc$concepts$sublattice(5:6)
A set of 4 concepts:
1: ({O1 [0.5], O2, O3, O4 [0.5]}, {P2, P4})
2: ({O1 [0.5], O4 [0.5]}, {P2, P3, P4})
3: ({O2, O3}, {P1 [0.5], P2, P4})
4: ({}, {P1, P2, P3, P4})
```

Some interesting sublattices appear when we consider only concepts fulfilling a given condition (e.g., a minimum support), using a command such as `fc$concepts$sublattice(fc$concepts$support() > 0.5)`.

Subconcepts, superconcepts, infimum and supremum

For a given concept (A, B), we can find all its subconcepts ((C_i, D_i) such that $(C_i, D_i) \leq (A, B)$) by using the `subconcepts()` functions. Analogously, the `superconcepts()` function can be used to compute the set of (C_i, D_i) such that $(A, B) \leq (C_i, D_i)$.

For instance, if we take a sample concept $L = \left(\left\{^{0.5}/\text{O1, O2, O3, }{}^{0.5}/\text{O4}\right\}, \{\text{P2, P4}\}\right)$, we can compute its subconcepts and superconcepts using:

```
> fc$concepts$subconcepts(L)
A set of 5 concepts:
1: ({O1 [0.5], O2, O3, O4 [0.5]}, {P2, P4})
2: ({O1 [0.5], O4 [0.5]}, {P2, P3, P4})
3: ({O2, O3}, {P1 [0.5], P2, P4})
4: ({O2 [0.5], O3 [0.5]}, {P1, P2, P4})
5: ({}, {P1, P2, P3, P4})
> fc$concepts$superconcepts(L)
A set of 2 concepts:
1: ({O1, O2, O3, O4}, {P2 [0.5], P4 [0.5]})
2: ({O1 [0.5], O2, O3, O4 [0.5]}, {P2, P4})
```

Also, we can define the infimum and the supremum of a set of concepts as the greatest common subconcept and the lowest common superconcept of all the given concepts, respectively. For a list of concepts, we can compute its infimum and supremum using the `supremum()` and `infimum()` methods of the `"ConceptLattice"` object. Additionally, irreducible elements in the lattice, with respect to join (supremum) and meet (infimum), can be computed for a given concept lattice with the methods `join_irreducibles()` and `meet_irreducibles()`.

```
> fc$concepts$meet_irreducibles()
A set of 5 concepts:
1: ({O1, O4}, {P2 [0.5], P3 [0.5], P4 [0.5]})
2: ({O1 [0.5], O4}, {P2 [0.5], P3, P4 [0.5]})
3: ({O1 [0.5], O2, O3, O4 [0.5]}, {P2, P4})
4: ({O2, O3}, {P1 [0.5], P2, P4})
5: ({O2 [0.5], O3 [0.5]}, {P1, P2, P4})
> fc$concepts$join_irreducibles()
A set of 5 concepts:
1: ({O1, O4}, {P2 [0.5], P3 [0.5], P4 [0.5]})
2: ({O1 [0.5], O4}, {P2 [0.5], P3, P4 [0.5]})
3: ({O1 [0.5], O4 [0.5]}, {P2, P3, P4})
4: ({O2, O3}, {P1 [0.5], P2, P4})
5: ({O2 [0.5], O3 [0.5]}, {P1, P2, P4})
```

These irreducible elements are essential since they constitute the basic elements with which the entire concept lattice can be reconstructed.

	M1	M2	M3	M4	M5
J1	×				
J2	×	×			
J3	×	×	×		
J4			×	×	
J5			×	×	×

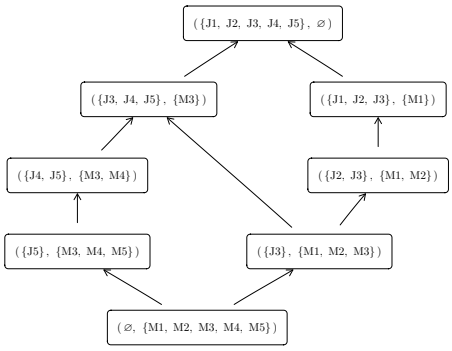


Table 7: Left: Standard context associated to the context in Table 1; M1 to M5 refer to the meet-irreducible elements of the formal context, and J1 to J5, to the join-irreducible ones. Right: concept lattice of the standard context, isomorphic to the one in Figure 1.

The standard context

The standard context has a concept lattice that is isomorphic to the one of the original context, so it encapsulates the same knowledge. With **fcaR** one can directly compute the standard context by using the `standardize()` function, for example, using `fc_std <- fc$standardize()` to save the new context to another variable.

This is a method that applies to a `"FormalContext"` object, but is closely related to its associated `"ConceptLattice"`, since the objects and attributes of the newly created standard context are the meet- and join- irreducible elements of the concept lattice.

The standard context for the example in Table 1 and its corresponding lattice are shown in Table 7.

Implications and logic

As mentioned earlier, a core problem in FCA is the computation of implications that hold in a formal context. The iterative fuzzy version of NEXTCLOSURE has been implemented in C and made accessible from **fcaR** thanks to an interface using **Rcpp**. As we have said, our package can manage classical implications from binary datasets. In order to build the Duquenne-Guigues basis of implications, by using the NEXTCLOSURE algorithm, the command `fc$find_implications()` is executed, and the result is stored in `fc$implications`.

The set of implications is stored as an object of class `"ImplicationSet"`, which can be inspected with the `print()` method or simply with `fc$implications`. In order to get a subset of the implications by providing its indexes, the standard R subsetting with '`[`' will create another `"ImplicationSet"` with exactly the rules required. For instance, `fc$implications[1:3]` gives:

```
Implication set with 3 implications.
Rule 1: {} -> {P2 [0.5], P4 [0.5]}
Rule 2: {P2 [0.5], P4} -> {P2}
Rule 3: {P2, P4 [0.5]} -> {P4}
```

These implications can be converted to LaTeX using the `to_latex()` method, giving:

$$\begin{array}{lrcl} 1: & \varnothing & \Rightarrow & \{^{0.5}/\mathrm{P2},\ ^{0.5}/\mathrm{P4}\} \\ 2: & \{^{0.5}/\mathrm{P2},\ \mathrm{P4}\} & \Rightarrow & \{\mathrm{P2}\} \\ 3: & \{\mathrm{P2},\ ^{0.5}/\mathrm{P4}\} & \Rightarrow & \{\mathrm{P4}\} \end{array}$$

On the other hand, one can `filter()` the implications to retrieve just those with specific attributes in the LHS and RHS. As before, the method returns a new `"ImplicationSet"` object. For example, to get the implications with attributes P1 and P2 in the LHS and attribute P4 in the RHS, the user can execute `fc$implications$filter(lhs = c('P1', 'P2'), rhs = 'P4')`.

Some quantities can be computed from a set of implications:

- *Cardinality*, that is, the number of implications, with `fc$implications$cardinality()`.
- *Size*, which is the cardinality of the LHS and RHS of *each* implication, interpreted as fuzzy sets (thus non-integer sizes can appear): `fc$implications$size()`.
- *Support*, the proportion of objects in the formal context whose set of attributes is a *superset* of the LHS of each implication: `fc$implications$support()`.

Application of the simplification logic

In **fcaR**, the simplification logic has been implemented and made accessible from an `"ImplicationSet"` object through method `apply_rules`.

The list of equivalence rules applicable to an `"ImplicationSet"` is stored in a `registry` object from the **registry** package by Meyer (2019).

This registry is called `equivalencesRegistry`, and one can inspect its contents by using:

```
> equivalencesRegistry$get_entry_names()
[1] "Composition"          "Generalization"       "Reduction"
[4] "Simplification"       "Right Simplification" "Reorder"
```

These names correspond to the methods added to the registry by default and are used to index those methods. Every method is accompanied by a description, so that we can see its definition:

```
> equivalencesRegistry$get_entry('Composition')
     method Composition
        fun <<function>>
description A -> B and A -> C equivalent to A -> BC
```

We can even use abbreviated names to refer to the method. For instance, we can use 'comp' instead of 'Composition' in the above command to obtain the information about the composition rule.

The registry's use enables the user to extend the functionality provided by the **fcaR** package. One can define and implement new equivalences rules and add them to the registry, and make them available to the `apply_rules()` method.

In order to add a new equivalence rule, we use the following:

```
> equivalencesRegistry$set_entry(method = 'Method name',
+                                fun = method_function,
+                                description = 'Method description')
```

where `method_function()` is the R function that computes the equivalences, and has the signature `function(LHS,RHS,attributes)`, where `LHS` and `RHS` are the sparse matrices defining the left-hand and right-hand sides of the implications, `attributes` is the character vector of attribute names, and returns the modified `LHS` and `RHS`. This mechanism has been used in the package to implement additional equivalence rules.

The user can decide which rules to remove redundancies will be applied and in which order:

```
> fc$implications$apply_rules(rules = c('reduction',
+                                       'comp',
+                                       'gener',
+                                       'simpl'))
```

These methods can be applied to binary and fuzzy implications.

Computation of closure and recommendations

One of the primary uses of a set of implications extracted from a formal context is computing the closure of a set of attributes, that is, the fuzzy set obtained by applying the given implications on the set of attributes.

The `closure()` method returns both the closure and the reduced `"ImplicationSet"` if `reduce = TRUE` and only the closure if `reduce = FALSE` (default). For instance, we can create a fuzzy set where the attribute `P2` is present with:

```
> A <- Set$new(attributes = fc$attributes, P2 = 1)
```

Then, we can compute its closure and the reduced set of implications doing:

```
> fc$implications$closure(A, reduce = TRUE)
$closure
{P2, P4}

$implications
Implication set with 2 implications.
Rule 1: {P3 [0.5]} -> {P3}
Rule 2: {P1 [0.5], P3} -> {P1}
```

6 Usage example. Fuzzy diagnostic system

In this section, a complete example of the use of **fcaR** on real-world problems is presented: Designing a diagnostic system from a formal context with (fuzzy) medical data.

The dataset for this section is provided and documented in the package.

In this example, the aim is to build an automated system using the **fcaR** package to perform medical diagnosis. We have focused on neurological pathologies since, in recent years, an increasing number of initiatives have appeared to share, curate, and study specific, prevalent brain pathologies. Among these pathologies, schizophrenia is of the highest interest, and public, curated repositories have been released.

The data source is SchizConnect (Wang et al., 2016), an online data repository integrating and mediating data from other schizophrenia-related databases, such as COBRE (Aine et al., 2017), which collect neuroimaging, psychological, neurological and clinical information. SchizConnect allows retrieving data about the patients that fulfil some conditions introduced as a query to the database. A subset of the COBRE dataset has been retrieved by querying SchizConnect for 105 patients with neurological and clinical symptoms. We also collected their corresponding diagnosis.

Among the clinical attributes in the dataset, one can find:

- *Calgary Depression Scale for Schizophrenia* (Addington et al., 1990), nine items (attributes) assessing the level of depression in schizophrenia, differentiating between positive and negative aspects of the disease.
- The *Simpson-Angus Scale* (Simpson and Angus, 1970), six items to evaluate Parkinsonism-like alterations related to schizophrenia in an individual.
- The *Structured Clinical Interview for DSM-III-R Personality Disorders* (First et al., 1997), with nine variables related to the presence of signs affecting personality.
- The diagnosis for each individual: it can be *schizophrenia strict* (abbreviated '`dx_ss`') or *other diagnosis* (abbreviated '`dx_other`', which includes schizoaffective and bipolar disorders). These diagnoses are mutually exclusive; thus, only one of them is assigned to each patient.

In summary, the dataset consists of the previous 30 attributes related to signs or symptoms and two attributes related to diagnosis. So the dataset has 105 objects (patients), and 32 attributes to explore. The symptom attributes are multi-valued.

For a given attribute (symptom), the available grades are *absent*, *minimal*, *mild*, *moderate*, *moderate severe*, *severe* and *extreme*. Thus, taking into account these scale levels, the attributes are coded as fuzzy and graded attributes with values 0, $1/6$, $1/3$, $1/2$, $2/3$, $5/6$ and 1, respectively. Since the symptom attributes are ordinal, for the sake of simplicity, they have been encoded as grades in the interval $[0, 1]$, just by mapping the lowest (*absent*) value to 0 and the greatest (*extreme*) to 1 and placing the remaining values equidistantly in the interval. In Ganter and Obiedkov (2016), other strategies of scaling for ordinal, nominal, or interval attributes are presented.

In **fcaR**, this dataset is exposed to the user with the name `cobre32`, and we can use it to create a fuzzy formal context (see Figure 3):

```
> fc <- FormalContext$new(cobre32)
```

Now, let us build a diagnosis system that employs the tacit knowledge present in the formal context. The objective is to define a function that takes a `"Set"` as input (with the clinical attributes of an individual) and returns the degree of the diagnosis attributes using the implications extracted from the formal context as an inference engine.

Next, we use the NEXTCLOSURE algorithm to extract implications and compute the set of concepts, using `fc$find_implications()`.

The concept lattice is quite big (14706 concepts); therefore, it cannot be plotted here for space and readability reasons. For this reason, we only plot a sublattice of small size in Figure 4.

There is an aggregate of 985 implications extracted. Let us compute the average cardinality of the LHS and the RHS of the extracted rules:

```
> colMeans(fc$implications$size())
     LHS      RHS
2.417597 1.954146
```

Note that our paradigm can deal with non-unit implications, that is, where there is more than one attribute in the RHS of the implication. This feature is an extension of what is usual in other paradigms, for example, in transactional databases.

We can use the *simplification logic* to remove redundancies and reduce the LHS and RHS size of the implications. The reason to do this is to decrease the computational cost of computing closures:

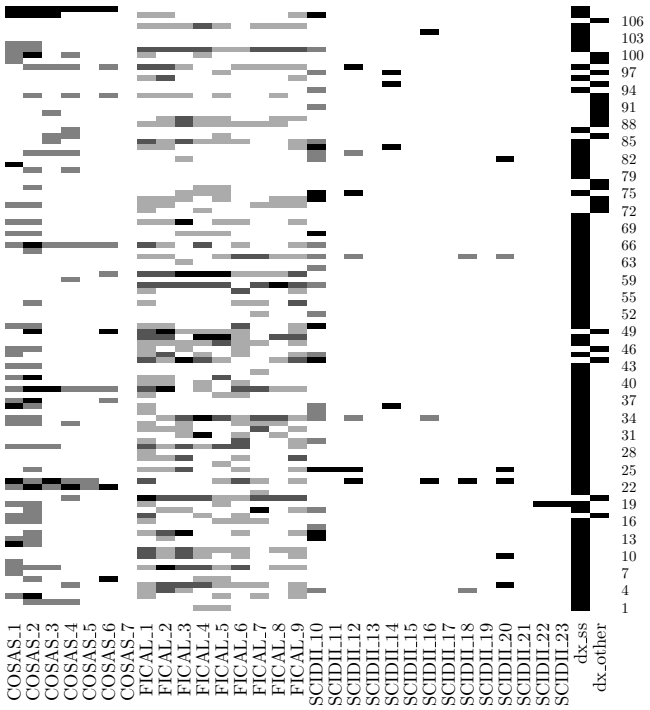


Figure 3: Formal context of the `cobre32` dataset. The default plot type allows us to identify patterns occurring in the attributes, such as the sparsity of the `SCIDII` variables. Additionally, we can see that the variables indicating diagnosis, `dx_ss` and `dx_other` in the last two columns, are binary and mutually exclusive.

```
> fc$implications$apply_rules(rules = c('simplification', 'rsimplification'))
> colMeans(fc$implications$size())
     LHS      RHS
1.998308 1.557191
```

We can see that the average cardinality of the LHS has been reduced from 2.418 to 1.998 and that the one of the RHS, from 1.954 to 1.557.

With the simplified implication set, we can build a recommender system by simply wrapping the `recommend()` method inside a function:

```
> diagnose <- function(S) {
+
+   fc$implications$recommend(S = S,
+                             attribute_filter =
+                               c('dx_ss', 'dx_other'))
+
+ }
```

This function can be applied to `"Set"`s that have the same attributes as those of the formal context. The `attribute_filter` argument specifies which attributes are of interest, in our case, the diagnosis attributes.

Let us generate some sets of attributes and get the recommendation (diagnosis) for each one:

```
> S1 <- Set$new(attributes = fc$attributes,
+               COSAS_1 = 1/2, COSAS_2 = 1, COSAS_3 = 1/2,
+               COSAS_4 = 1/6, COSAS_5 = 1/2, COSAS_6 = 1)
> diagnose(S1)
   dx_ss dx_other
       1        0
> S2 <- Set$new(attributes = fc$attributes,
+               COSAS_2 = 1, COSAS_6 = 1, FICAL_1 = 1/3, FICAL_3 = 1/3)
> diagnose(S2)
   dx_ss dx_other
```

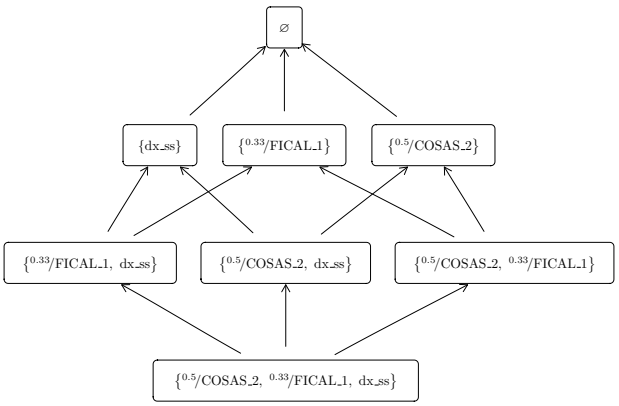


Figure 4: Hasse diagram for a sublattice of the `cobre32` formal context. Since the complete concept lattice is huge, with the Hasse diagram of a sublattice we can understand the relationship in a subset of the attributes. In this case, we can see how variables `COSAS_2` and `FICAL_1` are related to the variable `dx_ss` which indicates the schizophrenia diagnosis.

```
       0        0
> S3 <- Set$new(attributes = fc$attributes,
+               COSAS_4 = 2/3, FICAL_3 = 1/2, FICAL_5 = 1/2, FICAL_8 = 1/2)
> diagnose(S3)
   dx_ss dx_other
       0        1
```

These results mean that, for `S1`, the recommended diagnosis is schizophrenia strict, for `S2` there is not enough information, and the recommended diagnosis for `S3` is *other*, different from schizophrenia strict. For `S2`, maybe adding more attributes to it can help in obtaining a diagnosis.

One can inspect the reduced set of implications obtained after computing the closure for `S2`, simplify them and filter them to get the implications that can be applied if more attribute values are known for `S2`:

```
> cl <- fc$implications$closure(S2, reduce = TRUE)
> cl$implications$apply_rules(c('simp', 'rsimp', 'reorder'))
> cl$implications$filter(rhs = c('dx_ss', 'dx_other'),
+                        not_lhs = c('dx_ss', 'dx_other'), drop = TRUE)
Implication set with 12 implications.
Rule 1: {FICAL_5 [0.33]} -> {dx_other}
Rule 2: {FICAL_6 [0.33], FICAL_8 [0.33]} -> {dx_ss}
Rule 3: {SCIDII_18 [0.33]} -> {dx_ss}
Rule 4: {COSAS_1 [0.5], FICAL_8 [0.33]} -> {dx_ss}
Rule 5: {SCIDII_20 [0.33]} -> {dx_ss}
Rule 6: {SCIDII_16 [0.33]} -> {dx_ss}
Rule 7: {SCIDII_12 [0.33]} -> {dx_ss}
Rule 8: {FICAL_7 [0.33]} -> {dx_ss}
Rule 9: {FICAL_6 [0.33], SCIDII_10 [0.5]} -> {dx_ss}
Rule 10: {COSAS_3 [0.5]} -> {dx_ss}
Rule 11: {COSAS_1 [0.5]} -> {dx_ss}
Rule 12: {SCIDII_10 [0.5]} -> {dx_ss}
```

We can check that, for `S2`, if the presence of any symptom (of the LHS of the implications) is verified, then the implications above could directly tell us the diagnosis.

An extended version of the diagnosis system in this example has been presented in Cordero et al. (2020), where the **fcaR** package has been used to build a conversational recommender system based on fuzzy rules. In that work, the recommender system designed with the help of **fcaR** has obtained better results than those of the classical methods used in the area of recommendations.

7 Conclusions and future work

This work aims to present the first R package implementing the core methods in Formal Concept Analysis. This development aims to provide a helpful tool for the FCA community and other fields where knowledge discovery and retrieval play an essential role. Notably, the hybridization of FCA with other Machine Learning techniques and its application to data mining processes is well-known, making FCA an appealing tool for knowledge discovery.

This package provides **R6** classes and methods to manage datasets presented as formal contexts, use the derivation operators, work with the concepts (closed itemsets) or find and manage the basis of implications. Additionally, the **fcaR** package implements a logic to infer knowledge from sets of implications, allowing to compute closures and therefore, it can be used as the engine to build recommender systems and automated reasoning tools.

Another feature included in the package is providing graphical visualisations of the concept lattice that condenses the extracted knowledge.

The tool has been developed from two principles: *integration* with other packages and *extensibility*. First, the **arules** package is a *de facto* standard when working with transactional databases, closed itemsets and association rules in R. Since FCA's perspective can be seen as complementary to this, **fcaR** is able to import and export both the datasets and the implications from/to the analogous classes defined in **arules**. This way, the user of **arules** can benefit from the FCA tools in **fcaR**, including the simplification logic.

The extensibility of the **fcaR** is guaranteed by its design philosophy. First, the object-oriented programming paradigm allows extending the classes and methods to other datatypes, other kinds of formal contexts, or, for example, association rules. Second, using a registry for equivalence rules makes it easy to include new logic tools in the package's architecture without affecting existing code. Thus, users can extend the package's functionality by incorporating their methods and then testing and comparing them in a single framework.

Thus, **fcaR** implements a wide range of features. With the help of the included documentation and the comprehensive vignettes, any user can start analysing datasets with FCA tools quickly.

Regarding the low-level implementation, we have used sparse matrices as the primary internal data structure of the package since they represent a space- and cost-efficient storage. Also, when needed, the **fcaR** uses parallel computing and C routines to increase efficiency and tackle algorithmic bottlenecks.

As an example of use, we have used the implications extracted from a dataset to develop a recommender system to help diagnose schizoaffective disorders quickly.

The package is currently in stable development status. However, we devise future extensions in terms of new or improved algorithms or as new applications arise (in areas such as recommender systems, unsupervised learning or text mining).

We emphasize that **fcaR** is having a great acceptance. At the time of writing, it has now reached more than 20,000 downloads from CRAN.

Acknowledgments

This work has been partially supported by the projects TIN2017-89023-P (Spanish Ministry of Economy and Competitiveness, including FEDER funds), PGC2018-095869-B-I00 (Spanish Ministry of Science and Innovation), and the Junta de Andalucía project UMA2018-FEDERJA-001, co-funded by the European Regional Development Fund.

Bibliography

D. Addington, J. Addington, and B. Schissel. A depression rating scale for schizophrenics. *Schizophrenia research*, 3(4):247–251, 1990. [p354]

C. J. Aine, H. J. Bockholt, J. R. Bustillo, J. M. Cañive, A. Caprihan, C. Gasparovic, F. M. Hanlon, J. M. Houck, R. E. Jung, J. Lauriello, J. Liu, A. R. Mayer, N. I. Perrone-Bizzozero, S. Posse, J. M. Stephen, J. A. Turner, V. P. Clark, and V. D. Calhoun. Multimodal neuroimaging in schizophrenia: Description and dissemination. *Neuroinformatics*, 15(4):343–364, 2017. URL https://doi.org/10.1007/s12021-017-9338-9. [p354]

W. W. Armstrong. Dependency Structures of Data Base Relationships. In *IFIP Congress*, pages 580–583, 1974. [p344]

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2021. URL https://CRAN.R-project.org/package=Matrix. R package version 1.3-3. [p347]

P. Becker and J. H. Correia. The ToscanaJ suite for implementing conceptual information systems. In *Formal Concept Analysis*, pages 324–348. Springer Berlin Heidelberg, 2005. URL https://doi.org/10.1007/11528784_17. [p346]

R. Belohlávek and V. Vychodil. Attribute dependencies for data with grades I. *International Journal of General Systems*, 45(7-8):864–888, 2016. URL https://doi.org/10.1080/03081079.2016.1205711. [p341, 342]

R. Belohlávek and V. Vychodil. Attribute dependencies for data with grades II. *International Journal of General Systems*, 46(1):66–92, 2017. doi: 10.1080/03081079.2016.1205712. URL https://doi.org/10.1080/03081079.2016.1205712. [p341, 342]

R. Belohlávek, P. Cordero, M. Enciso, A. Mora, and V. Vychodil. Automated prover for attribute dependencies in data with grades. *International Journal of Approximate Reasoning*, 70:51–67, 2016. URL https://doi.org/10.1016/j.ijar.2015.12.007. [p342]

G. Birkhoff. *Lattice Theory*, volume 25. American Mathematical Soc., 1940. [p343]

W. Chang. *R6: Encapsulated Classes with Reference Semantics*, 2020. URL https://CRAN.R-project.org/package=R6. R package version 2.5.0. [p347]

P. Cordero, M. Enciso, A. Mora, and I. P. de Guzmán. SL_{FD} logic: Elimination of data redundancy in knowledge representation. In *IBERAMIA*, volume 2527 of *LNCS*, pages 141–150. Springer, 2002. URL https://doi.org/10.1007/3-540-36131-6_15. [p341, 344]

P. Cordero, M. Enciso, A. Mora, and M. Ojeda-Aciego. Computing minimal generators from implications: a logic-guided approach. In *CLA 2012*, volume 972 of *CEUR W.Proc.*, pages 187–198. CEUR-WS.org, 2012. URL http://ceur-ws.org/Vol-972/paper16.pdf. [p342]

P. Cordero, M. Enciso, D. López, and A. Mora. A conversational recommender system for diagnosis using fuzzy rules. *Expert Systems with Applications*, 154:113449, 2020. ISSN 0957-4174. URL https://doi.org/10.1016/j.eswa.2020.113449. [p356]

C. Demko and K. Bertet. GALACTIC: Galois lattices, concept theory, implicational systems and closures. a set of python3 packages for studying formal concept analysis. *Theoretical Computer Science*, 845:1–20, 2020. [p346]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL https://doi.org/10.18637/jss.v040.i08. [p345]

M. B. First, R. L. Spitzer, M. Gibbon, and J. B. Williams. *User's Guide for the Structured Clinical Interview for DSM-IV Axis I Disorders SCID-I: Clinician Version*. American Psychiatric Pub, 1997. [p354]

B. Ganter and S. A. Obiedkov. *Conceptual Exploration*. Springer, 2016. ISBN 978-3-662-49290-1. URL https://doi.org/10.1007/978-3-662-49291-8. [p342, 345, 354]

B. Ganter and R. Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999. ISBN 978-3-540-62771-5. URL https://doi.org/10.1007/978-3-642-59830-2. [p341]

J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences humaines*, 95:5–18, 1986. [p344]

M. Hahsler, B. Grun, and K. Hornik. arules - a computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, 14:1–25, 2005. URL http://dx.doi.org/10.18637/jss.v014.i15. [p346]

T. Hanika and J. Hirth. Conexp-Clj - a research tool for fca. *ICFCA (Supplements)*, 2378:70–75, 2019. [p346]

D. I. Ignatov. Introduction to formal concept analysis and its applications in information retrieval and related fields. *CoRR*, abs/1703.02819, 2017. [p341]

L. Jezková, P. Cordero, and M. Enciso. Fuzzy functional dependencies: A comparative survey. *Fuzzy Sets and Systems*, 317:88–120, 2017. URL https://doi.org/10.1016/j.fss.2016.06.019. [p342]

S. O. Kuznetsov. Machine learning and formal concept analysis. In *ICFCA 2004*, volume 2961 of *LNCS*, pages 287–312. Springer, 2004. URL https://doi.org/10.1007/978-3-540-24651-0_25. [p341]

D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983. ISBN 0-914894-42-0. [p345]

D. Meyer. *registry: Infrastructure for R Package Registries*, 2019. URL https://CRAN.R-project.org/package=registry. R package version 0.5-1. [p353]

A. Mora, M. Enciso, P. Cordero, and I. P. de Guzmán. An efficient preprocessing transformation for functional dependencies sets based on the substitution paradigm. In *CAEPIA 2003*, volume 3040 of *LNCS*, pages 136–146. Springer, 2003. URL https://doi.org/10.1007/978-3-540-25945-9_14. [p342]

A. Mora, P. Cordero, M. Enciso, I. Fortes, and G. Aguilera. Closure via functional dependence simplification. *International Journal of Computer Mathematics*, 89(4):510–526, 2012. URL https://doi.org/10.1080/00207160.2011.644275. [p342, 344, 345]

J. M. Moyano and L. Sanchez Ramos. RKEEL: using KEEL in R code. *FUZZ-IEEE 2016*, pages 257–264, July 2016. URL http://dx.doi.org/10.1109/FUZZ-IEEE.2016.7737695. [p346]

L. S. Riza, C. Bergmeir, F. Herrera, and J. M. Benítez. frbs: Fuzzy rule-based systems for classification and regression in R. *Journal of Statistical Software*, 65(6):1–30, 2015. URL https://doi.org/10.18637/jss.v065.i06. [p346]

E. Rodríguez Lorenzo, K. Bertet, P. Cordero, M. Enciso, and A. Mora. The direct-optimal basis via reductions. In *CLA 2014*, volume 1252 of *CEUR W.Proc.*, pages 145–156. CEUR-WS.org, 2014. URL http://ceur-ws.org/Vol-1252/cla2014_submission_18.pdf. [p342]

E. Rodríguez Lorenzo, K. V. Adaricheva, P. Cordero, M. Enciso, and A. Mora. From an implicational system to its corresponding d-basis. In *CLA 2015*, volume 1466 of *CEUR W.Proc.*, pages 217–228. CEUR-WS.org, 2015. URL http://ceur-ws.org/Vol-1466/paper18.pdf. [p342]

G. Simpson and J. Angus. A rating scale for extrapyramidal side effects. *Acta Psychiatrica Scandinavica*, 45(S212):11–19, 1970. [p354]

M. Trabelsi, N. Meddouri, and M. Maddouri. A new feature selection method for nominal classifier based on formal concept analysis. In *KES 2017*, volume 112 of *Procedia Computer Science*, pages 186–194. Elsevier, 2017. URL https://doi.org/10.1016/j.procs.2017.08.227. [p341]

K. Ushey, J. Allaire, and Y. Tang. *reticulate: Interface to Python*, 2020. URL https://CRAN.R-project.org/package=reticulate. R package version 1.18. [p345]

P. Valtchev, D. Grosser, C. Roume, and M. R. Hacene. Galicia: an open platform for lattices. In *ICCS'03*, pages 241–254. Citeseer, 2003. [p346]

L. Wang, K. I. Alpert, V. D. Calhoun, D. J. Cobia, D. B. Keator, M. D. King, A. Kogan, D. Landis, M. Tallis, M. D. Turner, S. G. Potkin, J. A. Turner, and J. L. Ambite. Schizconnect: Mediating neuroimaging databases on schizophrenia and related disorders for large-scale integration. *NeuroImage*, 124: 1155–1167, 2016. URL https://doi.org/10.1016/j.neuroimage.2015.06.065. [p354]

H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. URL https://doi.org/10.21105/joss.01686. [p345]

R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In *Ordered Sets*, pages 445–470. Springer, 1982. [p341]

Y. Xie, J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. ISBN 9781138359338. [p345]

S. A. Yevtushenko. System of data analysis Concept Explorer. In *Proc. 7th National Conference on Artificial Intelligence (KII'00)*, pages 127–134, 2000. [p346]

Pablo Cordero
Universidad de Málaga
Departamento de Matemática Aplicada
ETSI Telecomunicaciones, Campus de Teatinos
http://webpersonal.uma.es/de/pcordero/My_personal_web/Wellcome.html

ORCiD: 0000-0002-5506-6467
pcordero@uma.es

Manuel Enciso
Universidad de Málaga
Departamento de Lenguajes y Ciencias de la Computación
ETSI Informática, Campus de Teatinos
http://lcc.uma.es/~enciso/index_EN.html
ORCiD: 0000-0002-0531-4055
enciso@uma.es

Domingo López-Rodríguez
Universidad de Málaga
Departamento de Matemática Aplicada
ETSI Telecomunicaciones, Campus de Teatinos
https://dominlopez.netlify.app
ORCiD: 0000-0002-0172-1585
dominlopez@uma.es

Ángel Mora
Universidad de Málaga
Departamento de Matemática Aplicada
ETSI Informática, Campus de Teatinos
https://amorabonilla.github.io/
ORCiD: 0000-0003-4548-8030
amora@uma.es

FMM: An R Package for Modeling Rhythmic Patterns in Oscillatory Systems

by Itziar Fernández, Alejandro Rodríguez-Collado, Yolanda Larriba, Adrián Lamela, Christian Canedo and Cristina Rueda

Abstract This paper is dedicated to the R package **FMM** which implements a novel approach to describe rhythmic patterns in oscillatory signals. The frequency modulated Möbius (FMM) model is defined as a parametric signal plus a Gaussian noise, where the signal can be described as a single or a sum of waves. The FMM approach is flexible enough to describe a great variety of rhythmic patterns. The **FMM** package includes all required functions to fit and explore single and multi-wave FMM models, as well as a restricted version that allows equality constraints between parameters representing a priori knowledge about the shape to be included. Moreover, the **FMM** package can generate synthetic data and visualize the results of the fitting process. The potential of this methodology is illustrated with examples of such biological oscillations as the circadian rhythm in gene expression, the electrical activity of the heartbeat and the neuronal activity.

1 Introduction

Oscillations naturally occur in a multitude of physical, chemical, biological, and even economic and social processes. Periodic signals appear, for example, during the cell-cycle, in biological time-keeping processes, in human heartbeats, in neuronal signals, in light emissions from certain types of stars, or in business cycles in economics, among many others. Three features typically describe the periodic nature of the oscillatory motion: period, amplitude and phase. The period is the time required for one complete oscillation. Within a period, a sum of monocomponent models, characterized by the phase and amplitude parameters, can be used to describe the rhythmic pattern of a signal (Boashash, 2016). By varying the number of monocomponents and considering phase and amplitude parameters as fixed or variable, a large number of rhythmic signal representations can be found.

One of the most popular representations of oscillating signals is the Fourier decomposition (FD): a multicomponent representation with a fixed amplitude parameter. Its monocomponent version, the cosinor model (COS) (Cornelissen, 2014), is widely used, in particular in chronobiology, with acceptable results when a sinusoidal shape response within a period is expected. Due to its widespread use, many software utilities are available. Particularly in R, the estimation of a COS model can be performed using **cosinor** (Sachs, 2014) and **cosinor2** packages (Mutak, 2018). In addition, other packages from widely differing areas of knowledge have specific functions for fitting COS models. Such is the case of, for example, the function `CATCosinor` in the **CATkit** package (Gierke et al., 2018), which implements tools for periodicity analysis; the function `cosinor` in the **psych** package (Revelle, 2021), dedicated to personality and psychological research; or the function `cosinor` contained in a recent package, **card** (Shah, 2020), which is dedicated to the assessment of the regulation of cardiovascular physiology. Recently, it has also been implemented in other languages such as CosinorPy, a cosinor python package (Moskon, 2020). The COS model is easy to use and interpret with symmetrical patterns. However, asymmetric shapes are not captured properly by COS. When the waveform is nonsinusoidal, the use of multiple components analysis to fit a model consisting of a sum of several periodical functions is recommended. However, the multicomponent FD models, developed to provide flexibility from COS, often require the use of a large number of components resulting in serious overfitting issues.

In recent years, alternative methods, mostly nonparametric statistical methods, have been developed and used for analyzing rhythmicity, especially in biological data sets. Some very popular ones, such as the JTK_CYCLE (Hughes et al., 2010), wrongly assume that any underlying rhythms have symmetric waveforms. Others, such as RAIN (Thaben and Westermark, 2014), designed to detect more diverse wave shapes including asymmetric patterns, are not focused on modeling but on detecting rhythmic behavior in sets of data. Thus, they are not useful to describe the underlying oscillatory phenomena. The proliferation of methodology in this field has been accompanied by software developments. This is the case, for example, of the **DiscoRhythm** R package (Carlucci et al., 2020), very recently available on Bioconductor with a web interface based on the R **Shiny** platform (Chang et al., 2021). This tool allows four popular approaches to be used, including the COS model and JTK_CYCLE, to discover biological rhythmicity. Another recent example is the **circacompare** (Parsons et al., 2020), an R package implemented for modeling cosinusoidal curves by nonlinear regression. Hosted on GitHub, we can also find the **LimoRhyde** R package (`https://github.com/hugheylab/limorhyde`) for the differential analysis of rhythmic transcriptome data, based on fitting linear models (Singer and Hughey, 2019).

Motivated by the need for a flexible, interpretable and parametric methodology to fit rhythmic patterns, our research group recently proposed the frequency modulated Möbius (FMM) model (Rueda et al., 2019). The FMM is an additive nonlinear parametric regression model capable of adapting to nonsinusoidal shapes and whose parameters are easily interpretable. The single component model has been shown to successfully fit data as diverse as circadian clock signals, hormonal levels data or light data from distant stars. In addition, for more complex oscillatory signals, a multicomponent model of order m, denoted as FMM$_m$, which includes m single FMM components, can be used. This is, for example, the case for describing electrocardiography (ECG) signals. The FMM$_{ecg}$ signal, presented in Rueda et al. (2021b), is defined as the combination of five single FMM components. Another interesting area where the FMM approach has already shown its usefulness is in electrophysiological neuroscience. Specifically, we have proposed FMM methodology for modeling neuronal action potential (AP) curves, oscillating signals that measure the difference between the electrical potential inside and outside the cell (see Rueda et al., 2021c; Rodríguez-Collado and Rueda, 2021a). An FMM$_2$ model provides an accurate fitting for a single AP curve; whereas series of AP curves with similar repetitive spikes can be efficiently fitted by the FMM$_{ST}$ model, a restricted version of the multicomponent FMM model.

In this work we introduce the **FMM** package (Fernández et al., 2021), programmed in R and available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=FMM`. The package implements all required functions to fit and explore single and multicomponent FMM models, as well as a restricted multicomponent version. In addition, the **FMM** package provides functions to generate synthetic data and visualize the results of the fitted model. Furthermore, its use is illustrated in the aforementioned applications. The remainder of this paper is organized as follows: the next section provides a brief overview of both mono and multicomponent FMM models, as well as the FMM$_m$ model with equality constraints. The section follows is dedicated to the implementation details of the **FMM** package. After that, through a simulated example, the basic usage of the package is introduced, including the data generation and the fitting, as well as the visualization of the results. Then, the **FMM** package performance is shown through three application areas governed by oscillatory systems: chronobiology, ECG and neuroscience. Finally, a summary is provided.

2 Frequency modulated Möbius (FMM) model

FMM is a new approach to describe a great variety of rhythmic patterns in oscillatory signals as the composition of several additive components. In this section an overview of the FMM approach is provided. All the methodological details that justify the mathematical formulation of the FMM models are given in Rueda et al. (2019).

At the time point t, a single FMM wave is defined as $W(t;v) = A\cos(\phi(t;\alpha,\beta,\omega))$ where $v = (A,\alpha,\beta,\omega)'$, $A \in \Re^+$ represents the wave amplitude and,

$$\phi(t;\alpha,\beta,\omega) = \beta + 2\arctan\left(\omega\tan\left(\frac{t-\alpha}{2}\right)\right) \tag{1}$$

the wave phase. The phase angle ϕ of an FMM wave is defined using the Möbius link (see Downs and Mardia, 2002; Kato et al., 2008) rather than the linear link function as in the COS model. The Möbius link provides much more flexibility to describe nonsinusoidal patterns. Without loss of generality, we assume that the time point $t \in [0, 2\pi]$. Otherwise it can be transformed into $t' \in [t_0, T + t_0]$ by $t = \frac{(t'-t_0)2\pi}{T}$.

Each of the four parameters of an FMM wave characterizes some aspect of a rhythmic pattern. A describes the amplitude of the signal, while α, β and ω describe the wave phase. $\alpha \in [0, 2\pi]$ is a translation parameter and a wave location parameter in the real space, whereas $\beta \in [0, 2\pi]$ and $\omega \in [0, 1]$ describe the wave shape. To be precise, assuming $\alpha = 0$, the unimodal symmetric waves are characterized by values of β close to 0, π or 2π. When $\beta = \frac{\pi}{2}$ or $\beta = \frac{3\pi}{2}$, extreme asymmetric patterns are described. Moreover, a value of ω close to zero describes an extreme spiked wave and, as ω value increases, the pattern is increasingly smoother. When $\omega = 1$, a sinusoidal wave is described and the FMM model matches the COS model where $\varphi = \beta - \alpha$ is the acrophase parameter.

Two important features of a wave are the peak and trough, defined as the highest and lowest points above and below the rest position, respectively. In many applications, the peak and trough times could be very useful tools to extract practical information of a wave, since they capture important aspects of the dynamics. These two interesting parameters can be directly derived from the main parameters of

an FMM wave as,

$$t^U = \alpha + 2\arctan\left(\frac{1}{\omega}\tan\left(-\frac{\beta}{2}\right)\right) \tag{2}$$
$$t^L = \alpha + 2\arctan\left(\frac{1}{\omega}\tan\left(\frac{\pi-\beta}{2}\right)\right)$$

where t^U and t^L denote the peak and trough times, respectively.

Monocomponent FMM model

Let $X(t_i)$, $t_1 < t_2 < \cdots < t_n$ be the vector of observations. The monocomponent FMM model is defined as follows:

$$X(t_i) = M + W(t_i; v) + e(t_i), \quad i = 1, \ldots, n \tag{3}$$

where $M \in \Re$ is an intercept parameter describing the baseline level of the signal, $W(t_i; v)$ is an FMM wave, and it is assumed that the errors $e(t_i)$ are independent and normally distributed with zero mean and a common variance σ^2.

Estimation algorithm

A two-step algorithm to estimate monocomponent FMM model parameters is proposed. We now describe the substantial details of each stage of the algorithm.

Step 1: Initial parameter estimation. A two-way grid search over the choice of (α, ω) parameters is performed. For each pair of (α, ω) fixed values, the estimates for M, A and β are obtained by solving a least square problem as detailed below.

The model for a single FMM component can be written as:

$$X(t_i) = M + A\cos(t_i^* + \varphi) + e(t_i) \tag{4}$$

where $t_i^* = \alpha + 2\arctan\left(\omega\tan\left(\frac{t_i-\alpha}{2}\right)\right)$, $\varphi = \beta - \alpha$, and $e(t_i) \sim N(0, \sigma^2)$ for $i = 1, \ldots, n$.

Using trigonometric angle sum identity, the model can be rewritten as:

$$X(t_i) = M + \delta z_i + \gamma w_i + e(t_i) \tag{5}$$

where $\delta = A\cos(\varphi)$, $\gamma = -A\sin(\varphi)$, $z_i = \cos(t_i^*)$ and $w_i = \sin(t_i^*)$.

Since α and ω are fixed, the estimates for M, δ and γ are obtained by minimizing the residual sum of the squares (RSS),

$$RSS = \sum_{i=1}^{n}\left(X(t_i) - \left(\hat{M} + \hat{\delta} z_i + \hat{\gamma} w_i\right)\right)^2 \tag{6}$$

And the estimates for M, A and β are straightforward to derive as follows,

$$\hat{M} = \bar{X} - \hat{\delta}\sum_{i=1}^{n} z_i - \hat{\gamma}\sum_{i=1}^{n} w_i \tag{7}$$
$$\hat{A} = \sqrt{\hat{\delta}^2 + \hat{\gamma}^2} \tag{8}$$
$$\hat{\beta} = \alpha + \varphi \tag{9}$$

The best combination of (α, ω) values, with the lowest RSS, is retained and the corresponding estimates are the initial parameter estimation values.

Step 2: Optimization. In the second step, the Nelder-Mead optimization method (Nelder and Mead, 1965) is used to obtain the final FMM parameter estimates that minimize the RSS.

Multicomponent FMM model

A multicomponent FMM model of order m, denoted by FMM_m, is defined as

$$X(t_i) = M + \sum_{J=1}^{m} W(t_i; v_J) + e(t_i) \tag{10}$$
$$t_1 < t_2 < \cdots < t_n; i = 1, \ldots, n$$

where $W\left(t_i; v_J\right)$, hereinafter denoted by $W_J\left(t_i\right)$, is the Jth FMM wave and,

- $M \in \Re$
- $v_J = \left(A_J, \alpha_J, \beta_J, \omega_J\right)' \in \Re^+ \times [0, 2\pi] \times [0, 2\pi] \times [0, 1]; J = 1, \ldots, m$
- $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_m \leq \alpha_1$
- $\left(e\left(t_1\right), \ldots, e\left(t_n\right)\right)' \sim N_n\left(0, \sigma^2 I_n\right)$

Model adequacy

The goodness of fit of an FMM model is measured with the R^2 statistic that represents the proportion of the variance explained by the model out of the total variance, that is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(X\left(t_i\right) - \hat{X}\left(t_i\right)\right)^2}{\sum_{i=1}^{n}\left(X\left(t_i\right) - \bar{X}\right)^2} \tag{11}$$

where $\hat{X}\left(t_i\right)$ represents the fitted value at $t_i, i = 1, \ldots, n$.

Estimation algorithm

An iterative backfitting algorithm is proposed to derive estimates for the FMM parameters. Let $\hat{W}_J^{(k)}\left(t_i\right)$ denote the fitted values from the Jth FMM wave at $t_i, i = 1, ..., n$ in the kth iteration. The algorithm is structured as follows:

1. Initialize. Set $\hat{W}_1^{(0)}\left(t_i\right) = \cdots = \hat{W}_m^{(0)}\left(t_i\right) = 0$.
2. Backfitting step. For $J = 1, \ldots, m$, calculate

$$r_J^{(k)}\left(t_i\right) = X\left(t_i\right) - \sum_{I<J} \hat{W}_I^{(k)}\left(t_i\right) - \sum_{I>J} \hat{W}_I^{(k-1)}\left(t_i\right); I = 1, \ldots, m \tag{12}$$

   and fit a monocomponent FMM model to $r_J^{(k)}\left(t_i\right)$ obtaining $\hat{\alpha}_J^{(k)}, \hat{\beta}_J^{(k)}, \hat{\omega}_J^{(k)}$ and $\hat{W}_J^{(k)}\left(t_i\right)$.
3. Repeat the backfitting step until the stopping criterion is reached. The stopping criterion is defined as the difference between the explained variability in two consecutive iterations: $R_k^2 - R_{k-1}^2 \leq C$, where R_k^2 (defined in Equation 11) is the proportion of variance explained by the model in the kth iteration and C a constant.
4. $\hat{M}$ and $\hat{A}_J$ are derived by solving

$$\min_{M \in \Re; A_J \in \Re^+} \sum_{i=1}^{n}\left(X\left(t_i\right) - M - \sum_{J=1}^{m} A_J \cos\left(\hat{\phi}_J\left(t_i\right)\right)\right)^2 \tag{13}$$

   where $\hat{\phi}_J\left(t_i\right) = \phi\left(t_i; \hat{\alpha}_J, \hat{\beta}_J, \hat{\omega}_J\right)$ defined in Equation 1.

Restricted multicomponent FMM model

Modeling signals with repetitive shape-similar waves can be very useful in some applications (see Rodríguez-Collado and Rueda, 2021a). In order to obtain more efficient estimators, equality constraints are imposed on the β and ω parameters of an FMM$_m$ model. In particular, we add d blocks of restrictions:

$$\begin{aligned} \beta_1 &= \cdots = \beta_{m_1} & \omega_1 &= \cdots = \omega_{m_1} \\ \beta_{m_1+1} &= \cdots = \beta_{m_2} & \omega_{m_1+1} &= \cdots = \omega_{m_2} \\ &\cdots \\ \beta_{m_{d-1}+1} &= \cdots = \beta_{m_d} & \omega_{m_{d-1}+1} &= \cdots = \omega_{m_d} \end{aligned} \tag{14}$$

The parameter estimation problem is solved by an adaptation of the standard procedure.

FMM$_m$ estimation algorithm with restrictions on the β parameters

Given the unrestricted estimates obtained in step 3, the estimates for $\beta_1, \beta_{m_1+1}, \ldots, \beta_{m_{d-1}+1}$ under equality restrictions (Equation 14) are computed as follows:

$$\begin{aligned}
\hat{\beta}_J^* &= \text{angularMean}\left(\hat{\beta}_1, \ldots, \hat{\beta}_{m_1}\right) && J = 1, \ldots, m_1 \\
\hat{\beta}_J^* &= \text{angularMean}\left(\hat{\beta}_{m_1+1}, \ldots, \hat{\beta}_{m_2}\right) && J = m_1 + 1, \ldots, m_2 \\
&\ldots \\
\hat{\beta}_J^* &= \text{angularMean}\left(\hat{\beta}_{m_{d-1}+1}, \ldots, \hat{\beta}_{m_d}\right) && J = m_{d-1} + 1, \ldots, m_d
\end{aligned}$$

Then, the algorithm continues to the next step.

FMM$_m$ estimation algorithm with restrictions on the ω parameters

When constraints for the ω parameters are incorporated, the grid search for the different ω values is outside the backfitting loops. When the number of blocks is large, the estimation procedure can be computationally unaffordable. In order to reduce the execution time, a two-nested backfitting algorithm is proposed. In the outer backfitting loop, a block is fitted. In the inner loop, the FMM waves belonging to the same block are estimated. This procedure generates a close to optimal solution and is a less computationally expensive alternative.

3 FMM package: Implementation details

The **FMM** code makes use of the **doParallel** package (Corporation and Weston, 2020) to embed parallelization for the fitting process. Several utilities from the **ggplot2** (Wickham, 2016) and **RColorBrewer** (Neuwirth, 2014) packages are occasionally necessary for the visualization of the fitted models.

The implementation of **FMM** is divided into four main functionalities described in the next four sections: the fitting of the FMM models, the new `S4` object of class `"FMM"`, the graphical visualization of the fittings and the simulation of synthetic data.

Some general details about the functions contained in the **FMM** package are shown in Table 1.

Function	Description
Fitting function	
`fitFMM(vData,timePoints,nback,...)`	Estimates an FMM$_{nback}$ model to `vData` observed at `timePoints`.
Utility functions	
`plotFMM(objFMM,...)`	Graphically displays an object of class `"FMM"`.
`generateFMM(M,A,alpha,beta,omega,...)`	Simulates values from an FMM model with parameters ($M =$ `M`, $A =$ `A`, $\alpha =$ `alpha`, $\beta =$ `beta`, $\omega =$ `omega`).
`getFMMPeaks(objFMM,...)`	Estimates peak and trough times, together with signal values at those times, for each FMM wave.
`extractWaves(objFMM)`	Extracts individual contribution to the fitted values of each FMM wave.
Standard methods for objects of class `"FMM"`	
`summary()`, `show()`, `coef()`, `fitted()`	

Table 1: Summary of the fitting, utility functions and standard methods implemented in **FMM** package.

Fitting an FMM model

An FMM model can be fitted using the main function `fitFMM()`. The description and default values of its inputs arguments are shown in Table 2.

The fitting function `fitFMM()` requires the `vData` input argument, which contains the data to be fitted. Two other arguments can be used to control a basic fitting: `timePoints`, which contains the

specific time points of the single period; and nback, with the number of FMM components to be fitted. For some applications, such as the study of circadian rhythms, data are collected over multiple periods. This information is received by the fitFMM() function through the input argument nPeriods. When nPeriods>1, the FMM fitting is carried out by averaging the data collected at each time point across all considered periods.

Argument	Default value	Description
vData	no default value	A "numeric" vector containing the data to be fitted by an FMM model.
nPeriods	1	A "numeric" value specifying the number of periods at which vData is observed.
timePoints	NULL	A "numeric" vector containing the time points per period at which data is observed. When timePoints = NULL an equally spaced sequence from 0 to 2π will be assigned.
nback	1	A "numeric" value specifying the number of FMM components to be fitted.
betaOmegaRestrictions	1 : nback	An "integer" vector of length nback indicating which FMM waves are constrained to have equal β and ω parameters.
maxiter	nback	A "numeric" value specifying the maximum number of iterations for the backfitting algorithm.
stopFunction	alwaysFalse	Function to check the stopping criterion for the backfitting algorithm.
lengthAlphaGrid	48	A "numeric" value specifying the grid resolution of the parameter α.
lengthOmegaGrid	24	A "numeric" value specifying the grid resolution of the parameter ω.
numReps	3	A "numeric" value specifying the number of times (α, ω) parameters are refined.
showProgress	TRUE	TRUE to display a progress indicator on the console.
showTime	FALSE	TRUE to display execution time on the console.
parallelize	FALSE	TRUE to use parallelized procedure to fit a FMM model.
restrExactSolution	FALSE	TRUE to obtain the optimal solution for the restricted fitting.

Table 2: Description of the input arguments of the fitFMM() function and their default values.

There are three key issues in the fitting process: the grid search of the pair (α, ω) to solve the estimation problem of a single FMM wave, the backfitting algorithm used for the estimation of the multicomponent models, and the incorporation of restrictions on β and ω parameters. Each of these issues is controlled by several arguments described below.

- **Grid search of the pair** (α, ω). The lengthAlphaGrid and lengthOmegaGrid arguments are used to set the grid resolution by specifying the number of equally spaced α and ω values. Thus, the objective function will be evaluated a total number of (lengthAlphaGrid)$\times$(lengthOmegaGrid) times, so when both arguments are large, the computational demand can be high. By reducing the size of the sequences of the α and ω parameters, the algorithm will be computationally more efficient. However, it may fail to obtain an accurate estimation if the grid resolution is too sparse. An implemented option to fine-tune the estimation of the parameters is to repeat the fitting process a numReps of times, in such a way that, at each repetition, a new two-dimensional grid of (α, ω) points is created around the previous estimates. In addition, the parallelize argument specifies whether a parallel processing implementation is used.
- **Backfitting algorithm**. The argument maxiter sets the maximum number of backfitting iterations. Through the argument stopFunction, it is possible to set a stopping criterion. Two criteria have been implemented as stop functions in this package. When stopFunction = alwaysFalse,

`maxiter` iterations will be forced. If `stopFunction = R2()`, the algorithm will be stopped when the difference between the explained variability in two consecutive iterations is less than a value pre-specified in the `difMax` argument of `R2()` function.

- **Restrictions**. The argument `betaOmegaRestrictions` sets the equality constraints for the β and ω parameters. For the unrestricted case, `betaOmegaRestrictions = 1:nback`. To add restrictions, `"integer"` vectors of length m can be passed to this argument, so that positions with the same numeric value correspond to FMM waves whose parameters, β and ω, are forced to be equal. Since restricted fitting can be computationally intensive, a two-nested backfitting algorithm can be used for the estimation of ω parameters when the argument `restrExactSolution = FALSE`.

Object of class `"FMM"`

The `fitFMM()` function outputs an `S4` object of class `"FMM"` which contains the slots presented in Table 3.

Slot	Description
`timePoints`	A `"numeric"` vector containing the time points for each data point if one single period is observed.
`data`	A `"numeric"` vector containing the data to be fitted to an FMM model. Data could be collected over multiple periods.
`summarizedData`	A `"numeric"` vector containing the summarized data at each time point across all considered periods.
`nPeriods`	A `"numeric"` value containing the number of periods in `data`.
`fittedValues`	A `"numeric"` vector of the fitted values by the FMM model.
`M`	A `"numeric"` value of the estimated intercept parameter M.
`A`	An m-element `"numeric"` vector of the estimated FMM wave amplitude parameter(s) A.
`alpha`	An m-element `"numeric"` vector of the estimated FMM wave phase translation parameter(s) α.
`beta`	An m-element `"numeric"` vector of the estimated FMM wave skewness parameter(s) β.
`omega`	An m-element `"numeric"` vector of the estimated FMM wave kurtosis parameter(s) ω.
`SSE`	A `"numeric"` value of the residual sum of squares values.
`R2`	An m-element `"numeric"` vector specifying the explained variance by each of the fitted FMM components.
`nIter`	A `"numeric"` value containing the number of iterations of the backfitting algorithm.

Table 3: Summary of the slots of the `S4` object of class `"FMM"` resulting from fitting an FMM model with m components.

The standard methods implemented for the class `"FMM"` include the functions `summary()`, `show()`, `coef()` and `fitted()`. These methods display relevant information of the FMM fitting, and provide the estimated parameters and fitted values. In addition, two more specific functions have been implemented. Through the `extractWaves()` function, the individual contribution of each FMM wave to the fitted values can be extracted. Finally, the location of the peak and trough of each FMM wave, as well as the value of the signal at these time points, can be estimated using the `getFMMPeaks()` function. The required argument of all these methods and functions is an object of the class `"FMM"`. Particularly, `getFMMPeaks()` has an optional argument: `timePointsIn2pi`, that forces the peak and trough locations to be returned into the interval from 0 to 2π when it is `TRUE`.

Plotting FMM models

The **FMM** package includes the function `plotFMM()` to visualize the results of an FMM fit. The arguments of this function are summarized in Table 4.

An object of class `"FMM"` can be plotted in two ways (see Figure 1). The default graphical representation will be a plot on which original data (as points) and the fitted signal (as a line) are plotted together (left panel in Figure 1). The other possible representation is a component plot for displaying each centered FMM wave separately (right panel in Figure 1). Set the boolean argument `components`

Argument	Default value	Description
objFMM	no default value	The object of class "FMM" to be plotted.
components	FALSE	TRUE to display a plot of components.
plotAlongPeriods	FALSE	TRUE to plot more than 1 period.
use_ggplot2	FALSE	TRUE to plot with **ggplot2** package.
legendInComponentsPlot	TRUE	TRUE to indicate if a legend should be plotted in the component plot.
textExtra	empty string	Extra text to be added to the title of the plot.

Table 4: Description of the input arguments of the plotFMM() function and their default values.

= TRUE to show a component plot. When legendInComponentsPlot = TRUE, a legend appears at the bottom of the component plot to indicate the represented waves. The argument textExtra allows an extra text to be added to the title of both graphical representations.

As mentioned above, in some cases, data are collected from different periods. All periods can be displayed simultaneously on the default plot using plotAlongPeriods = TRUE. For the component plot, this argument is ignored.

The argument use_ggplot2 provides a choice between building the plot using base R **graphics** or **ggplot2** packages. By default, the **graphics** package is used. When use_ggplot2 = TRUE, a more aesthetic and customizable plot is created using the **ggplot2** package.

Simulating data from an FMM model

Data from an FMM model can be easily simulated using the function generateFMM() of the package **FMM**. All input arguments of this function are shown in Table 5, along with a short description and their default values.

Argument	Default value	Description
M	no default value	Value of the intercept parameter M.
A	no default value	Vector of the FMM wave amplitude parameter A.
alpha	no default value	Vector of the FMM wave phase translation parameter α.
beta	no default value	Vector of the FMM wave skewness parameter β.
omega	no default value	Vector of the FMM wave kurtosis parameter ω.
from	0	Initial time point of the simulated data.
to	2π	Final time point of the simulated data.
length.out	100	Desired length of the simulation.
timePoints	seq(from,to,length = length.out)	Time points at which the data will be simulated.
plot	TRUE	TRUE when the simulated data should be drawn on a plot.
outvalues	TRUE	TRUE when the numerical simulation should be returned.
sigmaNoise	0	Standard deviation of the Gaussian noise to be added.

Table 5: Description of the input arguments of the generateFMM() function and their default values.

The main arguments of this function are M, A, alpha, beta and omega, whereby the values of the FMM model parameters are passed to the function. All these arguments are "numeric" vectors of length m, except M, which has length 1. Longer and smaller vectors will be truncated or replicated as appropriate.

By default, the data will be simulated at a sequence of 100 equally spaced time points from 0 to 2π. The arguments from, to and length.out control such sequences. The sequence can also be manually

set using the argument timePoints, in which case from, to and length.out will be ignored.

The user can add a Gaussian noise by argument sigmaNoise. A positive "numeric" value sets the corresponding standard deviation of the Gaussian noise to be added. To create the normally distributed noise, the rnorm() function is used.

The arguments plot and outvalues, both boolean values, determine the output of the generateFMM() function. When outvalues = TRUE, a "list" with input parameters, time points and simulated data is returned. These elements are named input, t and y, respectively. In addition, a scatter plot of y against t can be drawn by setting plot = TRUE.

4 Basic usage of the FMM package

The example below, based on FMM synthetic data, illustrates the basic uses and capabilities of the functions implemented in the **FMM** package. A set of 100 observations is simulated from an FMM_4 model with intercept parameter $M = 3$, amplitude parameters: $A_1 = 4$, $A_2 = 3$, $A_3 = 1.5$ and $A_4 = 1$, and phase translation parameters: $\alpha_1 = 3.8$, $\alpha_2 = 1.2$, $\alpha_3 = 4.5$ and $\alpha_4 = 2$. With regard to the shape parameters, pairs of waves are equal. Specifically, the shape parameters satisfy:

$$\beta_1 = \beta_2 = 3 \qquad \omega_1 = \omega_2 = 0.1$$
$$\beta_3 = \beta_4 = 1 \qquad \omega_3 = \omega_4 = 0.05$$

The standard deviation of the error term is set at $\sigma = 0.3$. We use the function generateFMM() to simulate this data set. A set.seed() statement is used to guarantee the reproducibility of the results.

```
> library("FMM")
> set.seed(1115)
> rfmm.data <-generateFMM(M = 3, A = c(4,3,1.5,1), alpha = c(3.8,1.2,4.5,2),
+                         beta = c(rep(3,2),rep(1,2)),
+                         omega = c(rep(0.1,2),rep(0.05,2)),
+                         plot = FALSE, outvalues = TRUE,
+                         sigmaNoise = 0.3)
```

The estimation of an FMM_4 can be performed by setting nback = 4 in the fitting function fitFMM(). The betaOmegaRestrictions parameter allows a wide variety of shape restrictions to be incorporated into the fitting procedure. In this example, to impose the shape restricticton on the fitting process, we use betaOmegaRestrictions = c(1,1,2,2).

```
> fit.rfmm <- fitFMM(vData = rfmm.data$y, timePoints = rfmm.data$t, nback = 4,
+                    betaOmegaRestrictions = c(1, 1, 2, 2))
|-------------------------------------------------|
|=================================================|
 Stopped by reaching maximum iterations (4 iteration(s))
```

The results are displayed by the function summary():

```
> summary(fit.rfmm)

Title:
FMM model with 4 components

Coefficients:
M (Intercept): 3.1661
                  A  alpha   beta  omega
FMM wave 1:  4.0447 3.8048 3.0238 0.0930
FMM wave 2:  3.1006 1.1956 3.0238 0.0930
FMM wave 3:  1.6069 4.5228 1.0145 0.0427
FMM wave 4:  1.1194 1.9788 1.0145 0.0427

Peak and trough times and signals:
             t.Upper Z.Upper t.Lower Z.Lower
FMM wave 1:   0.6741  5.3198  4.9354 -2.7565
FMM wave 2:   4.3482  3.4702  2.3263 -2.1742
FMM wave 3:   1.5345 -1.2330  1.3338 -4.1527
FMM wave 4:   5.2737 -1.7005  5.0730 -3.7565
```

```
Residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.719769 -0.162649  0.007025  0.000000  0.160127  0.904218

R-squared:
Wave 1 Wave 2 Wave 3 Wave 4  Total
0.5049 0.3906 0.0531 0.0276 0.9761
```

The FMM wave parameter estimates, as well as the peak and trough times, together with the signal values at those times, are presented in tabular form, where each row corresponds to a component and each column to an FMM wave parameter. As part of the summary, a brief description of the residuals, the proportion of variance explained by each FMM component and by the global model are also shown. The `summary()` output can be assigned to an object to get a `"list"` of all the displayed results.

Other options to return the results are the functions `coef()`, `getFMMPeaks()` and `resid()`. The first two return a `"list"` similar to those obtained with `summary()`. The `resid()` method can be used to obtain the complete residuals vector. In addition, the fitted values can be extracted by the function `fitted()`, which returns a `"data.frame"` with two columns: time points and fitted values.

The FMM plots can be generated in the R **graphics** or **ggplot2** packages. In the code example given below, we use `use_ggplot2 = TRUE` to build Figure 1 based on **ggplot2**. The use of **ggplot2** makes it easier to customize our plots and modify features, such as scales, margins, axes, etc. In Figure 1, the two possible FMM plots are arranged via the `grid.arrange()` function of the **gridExtra** package (Auguie, 2017).

```
> library("RColorBrewer")
> library("ggplot2")
> library("gridExtra")
> # Plot the fitted FMM model
> titleText <- "Simulation of four restricted FMM waves"
> defaultrFMM2 <- plotFMM(fit.rfmm, use_ggplot2 = TRUE, textExtra = titleText) +
+                 theme(plot.margin=unit(c(1,0.25,1.3,1), "cm")) +
+                 ylim(-5, 6)
> comprFMM2 <- plotFMM(fit.rfmm, components=TRUE, use_ggplot2 = TRUE,
+                      textExtra = titleText) +
+              theme(plot.margin=unit(c(1,0.25,0,1), "cm")) +
+              ylim(-5, 6) +
+              scale_color_manual(values = brewer.pal("Set1",n = 8)[3:6])
> grid.arrange(defaultrFMM2, comprFMM2, nrow = 1)
```

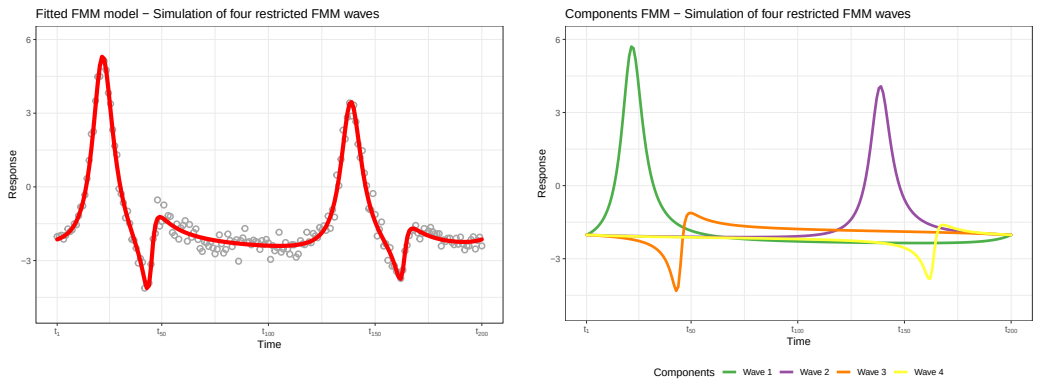


Figure 1: Graphical representation of the estimated restricted FMM$_4$ signal with $\beta_1 = \beta_2, \omega_1 = \omega_2$ and $\beta_3 = \beta_4, \omega_3 = \omega_4$ constraints. A scatter plot of the simulated data along with the fitted signal is displayed on the left (default plot). The component plot is shown on the right.

5 Real data analysis using the FMM package

This section illustrates the use of the **FMM** package on the analysis of real signals from chronobiology, electrocardiography and neuroscience. To do this, the package includes four real-world data sets in `RData` format which are described in the following sections.

Example 1: Chronobiology

Chronobiology studies ubiquitous daily variations found in nature and in many aspects of the physiology of human beings, such as blood pressure or hormone levels (Mermet et al., 2017). These phenomena commonly display signals with oscillatory patterns that repeat every 24 hours, usually known as circadian rhythms. In particular, circadian gene expression data have been deeply analyzed in the literature as they regulate the vast majority of molecular rhythms involved in diverse biochemical and cellular functions, see among others Zhang et al. (2014), Cornelissen (2014) and Larriba et al. (2020).

The **FMM** package includes a data set called `mouseGeneExp` that contains expression data of the *Iqgap2* gene from mouse liver. The liver circadian database is widely extended in chronobiology since the liver is a highly rhythmic organ with moderate levels of noise (Anafi et al., 2017; Larriba et al., 2018, 2020). The complete database is freely available at NCBI GEO (`http://www.ncbi.nlm.nih.gov/geo/`), with GEO accession number GSE11923. Gene expression values are given along 48 hours with a sampling frequency of 1 hour/2 days. Hence, data are collected along two periods, and an FMM_1 model is fitted to the *Iqgap2* average expressed values as follows:

```
> data("mouseGeneExp", package = "FMM")
> fitGene <- fitFMM(vData = mouseGeneExp, nPeriods = 2, nback = 1, showProgress = FALSE)
> summary(fitGene)

Title:
FMM model with 1 components

Coefficients:
M (Intercept): 10.1508
                  A  alpha   beta  omega
FMM wave 1:  0.4683 3.0839 1.5329 0.0816

Peak and trough times and signals:
             t.Upper Z.Upper t.Lower Z.Lower
FMM wave 1:   0.1115 10.6191  6.0686  9.6825

Residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-9.751e-02 -3.490e-02  2.269e-03 -1.530e-06  2.670e-02  1.890e-01

R-squared:
[1] 0.8752
```

The behavior of the FMM versus COS model to describe this asymmetric pattern has been compared in terms of R^2. The FMM model clearly outperforms the COS one with an R^2 of 0.8752 and 0.2835, respectively. In addition, a difference of 4.73 hours in peak time estimation between both models is observed, the FMM peak estimate being much more reliable, as is shown in Figure 2.

Example 2: Electrocardiography

ECG records the periodic electrical activity of the heart. This activity represents the contraction and relaxation of the atria and ventricle, processes related to the crests and troughs of the ECG waveform. Heartbeats are decomposed into five fundamental waves, labelled as P, Q, R, S and T, corresponding to the different phases of the heart's electric activity. The main features used in medical practice for cardiovascular pathology diagnosis are related to the location and amplitudes of these waves, and, of them, those labeled as P, R and T are of particular interest (Bayes de Luna, 2007). Standard ECG signals are registered using twelve leads, calculated from different electrode locations. Lead II is the reference signal, as it usually provides a good view of the main ECG waves (Meek and Morris, 2002).

The **FMM** package includes the analysis of a typical ECG heartbeat from the QT database (Laguna et al., 1997). This recording, from the subject *sel100*, belongs to the *Normal* category, regarding Physionet's pathology classification (Goldberger et al., 2000). The data illustrate the voltage of the heart's electric activity, measured in mV, along the heartbeat with a sampling frequency of $250Hz$. Specifically, the ECG signal from lead II in the fifth of the thirty annotated heartbeats is analysed. Recordings are publicly available on (`http://www.physionet.org`). Data are saved as `ecgData` in the package. For an ECG heartbeat, an FMM_{ecg}, a fifth order multicomponent FMM model can be fitted with the instruction:

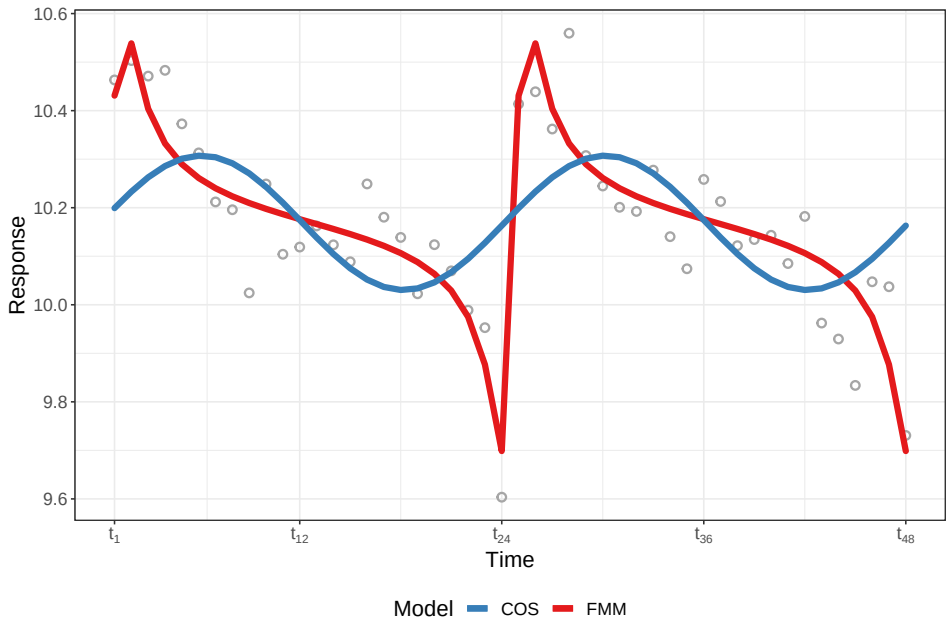


Figure 2: *Iqgap2* gene expression data along two periods (grey dots); FMM (red line) and COS (blue line) fitted signals.

```
> data("ecgData", package = "FMM")
> fitEcg <- fitFMM(ecgData, nback = 5, showProgress = FALSE)
> summary(fitEcg)

Title:
FMM model with 5 components

Coefficients:
M (Intercept): 5.2717
                  A  alpha   beta  omega
FMM wave 1:  0.6454 5.5151 3.2926 0.0325
FMM wave 2:  0.0994 4.4203 3.7702 0.1356
FMM wave 3:  0.2443 5.3511 0.6636 0.0323
FMM wave 4:  0.3157 5.5919 4.8651 0.0126
FMM wave 5:  0.0666 1.7988 2.1277 0.1632

Peak and trough times and signals:
             t.Upper Z.Upper t.Lower Z.Lower
FMM wave 1:   2.3686  6.2370  3.1841  4.7241
FMM wave 2:   1.1905  4.9487  2.0693  4.6897
FMM wave 3:   2.3965  6.0828  2.1872  4.5551
FMM wave 4:   2.4210  5.7933  2.4719  4.7175
FMM wave 5:   5.1212  4.8646  4.3689  4.7228

Residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.0690885 -0.0095597 -0.0001127  0.0000000  0.0098533  0.0623569

R-squared:
Wave 1 Wave 2 Wave 3 Wave 4 Wave 5  Total
0.7645 0.0920 0.0581 0.0493 0.0278 0.9918
```

It is worth noting that the **FMM** package not only provides ECG signal-fitting (the left hand panel in Figure 3), but it also does wave decomposition and fiducial mark annotations on the desired waves (the right hand panel in Figure 3). It is clearly visible how the specific shapes of the five main waves contribute to drawing and explaining the lead II ECG waveform from the *Normal* morphology. See Rueda et al. (2021b) for a complete review of FMM$_{ecg}$.

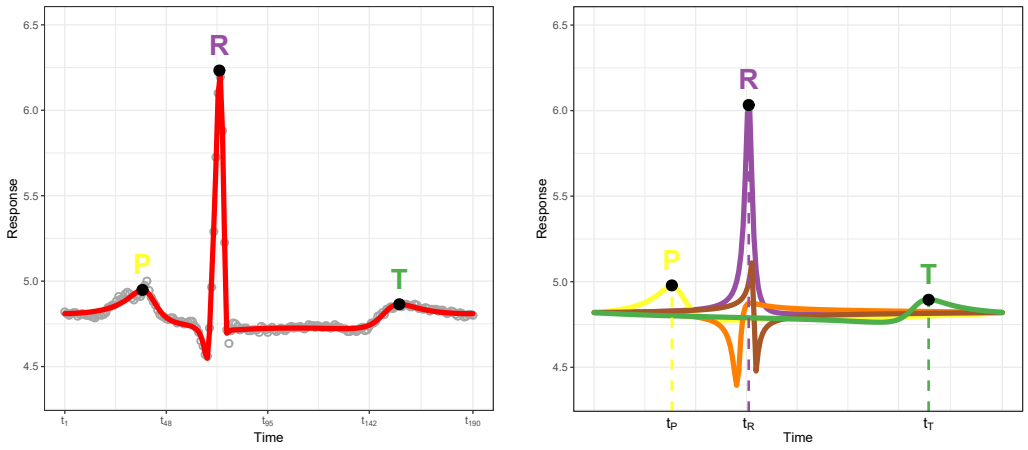


Figure 3: FMM$_{ecg}$ performance on a single beat from patient *sel100* from the QT database. Left: Data (grey dots) and FMM fitting (red line). Black dots locate the P, R and T fiducial marks. Right: ECG decomposition on P(orange), Q (purple), R (green), S (yellow) and T (blue) waves. Dash lines indicate P, R and T peak times.

Example 3: Neuroscience

Single AP curve

The study of the electrophysiological activity of neurons is one of the main research branches in neuroscience. The AP curves are oscillatory signals that serve as basic information units between neurons. They measure the electrical potential difference between inside and outside the cell due to an external stimulus. Gerstner et al. (2014) can serve as a basic reference for electrophysiological neuroscience. Recently, the shape and other features of the AP have been used in problems such as spike sorting (Rácz et al., 2020; Souza et al., 2019; Caro-Martín et al., 2018) or neuronal cell type classification (Teeter et al., 2018; Gouwens et al., 2019; Mosher et al., 2020; Rodríguez-Collado and Rueda, 2021b).

The package includes an example of a neuronal AP. The data were simulated with the renowned Hodgkin-Huxley model, first presented in Hodgkin and Huxley (1952), which is defined as a system of ordinary differential equations and has been used in a wide array of applications, as it successfully describes the neuronal activity in various organisms. The simulation has been done using a modified version of the python package NeuroDynex available at Gerstner et al. (2014). More concretely, a short square stimulus of $12\mu A$ has been applied to the neuron. The data can be accurately fitted by an FMM$_2$ model as follows:

```
> data("neuronalSpike", package = "FMM")
> fitSingleAP <- fitFMM(neuronalSpike, nback = 2, showProgress = FALSE)
> summary(fitSingleAP)

Title:
FMM model with 2 components

Coefficients:
M (Intercept): 44.9474
                    A  alpha   beta  omega
FMM wave 1:  52.9014 4.4160 3.0606 0.0413
FMM wave 2:  18.5046 4.6564 4.9621 0.0322

Peak and trough times and signals:
             t.Upper  Z.Upper t.Lower  Z.Lower
FMM wave 1:   1.2777 110.8361  5.9669  -2.5002
FMM wave 2:   1.4319  36.9084  1.5649 -16.2572

Residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
```

```
-14.3012  -1.0038   0.7472   0.0000   1.3230  24.8618

R-squared:
Wave 1 Wave 2  Total
0.9064 0.0604 0.9669
```

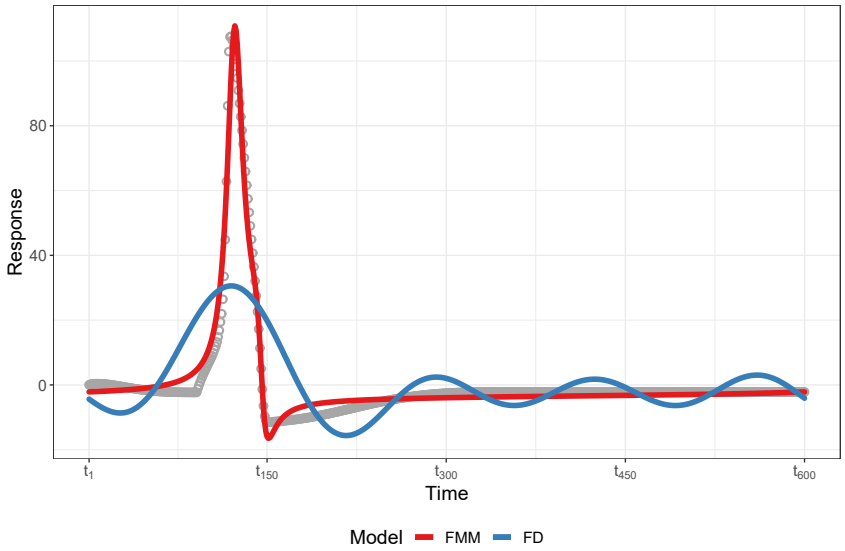


Figure 4: Neuronal AP simulated with the Hodgkin-Huxley model (parameters: $C = 1, g_{Na} = 260, g_K = 30, g_L = 0.31, V_K = -12, V_{Na} = 115, V_L = 10.6, \widetilde{a}_n = 1.15, \widetilde{b}_n = 0.85, \widetilde{a}_m = 0.9, \widetilde{b}_m = 1.3, \widetilde{a}_h = 1, \widetilde{b}_h = 1$ and applying a current of $12\mu A$ for 1 millisecond) and the estimated FMM_2 signal in red. An FD model of the same number of degree of freedom has been fitted and plotted in blue.

The goodness of fit of the FMM_2 model can be ascertained in Figure 4. For comparison purposes, an FD model has been fitted with the same number of degrees of freedom. While the FD attains an $R^2 = 0.3926$, the FMM model achieves a better fit with $R^2 = 0.9669$.

AP train

Multiple AP curves, denominated spike or AP train, are usually observed as the response to a stimulus. Various models, such as the widely used leaky-and-fire models (Lynch and Houghton, 2015), cut the signal into segments, each one containing an AP curve. Some authors suggest cutting the signal into even segments (Gerstner et al., 2014). However, the length of the segments turns out to be significantly different between different types of neurons, as explained in Teeter et al. (2018), and unequal data segments can lessen the utility of some approaches. An important aspect to take into account is that the shape of the APs in the spike train is considered to be similar and, consequently, a restricted FMM model can accurately fit the entire signal.

The **FMM** package includes the data of a spike train composed of three AP curves. The proposed model for use with these data is an FMM_{ST} model, as defined in Rodríguez-Collado and Rueda (2021a). Each AP is modeled by two components. The β and ω parameters are constrained between AP curves. The code below fits the model.

```
> data("neuronalAPTrain", package = "FMM")
> nAPs <- 3; restriction <- c(rep(1,nAPs),rep(2,nAPs))
> fitAPTrain<-fitFMM(neuronalAPTrain, nback = nAPs*2,
                     betaRestrictions = restriction,
                     omegaRestrictions = restriction,
                     showProgress = FALSE, parallelize=TRUE)
> summary(fitAPTrain)

Title:
FMM model with 6 components

Coefficients:
M (Intercept): 135.4137
```

```
                  A  alpha   beta  omega
FMM wave 1:  51.7069 6.1358 2.8172 0.0384
FMM wave 2:  52.0915 1.7541 2.8172 0.0384
FMM wave 3:  51.1140 4.2319 2.8172 0.0384
FMM wave 4:  20.3725 4.4778 4.8637 0.0552
FMM wave 5:  19.2429 1.9981 4.8637 0.0552
FMM wave 6:  19.6748 0.0973 4.8637 0.0552

Peak and trough times and signals:
            t.Upper  Z.Upper t.Lower  Z.Lower
FMM wave 1:  3.0067 111.4319  2.5332  -1.2051
FMM wave 2:  4.9082 111.7323  4.4347  -1.4607
FMM wave 3:  1.1028 111.2700  0.6293  -0.1561
FMM wave 4:  1.2077  58.5986  1.4310 -14.2508
FMM wave 5:  5.0113  58.5537  5.2345 -13.3010
FMM wave 6:  3.1104  58.4889  3.3337 -13.7041

Residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-14.8618  -1.4929   0.5029   0.0000   1.6021  19.1978

R-squared:
Wave 1 Wave 2 Wave 3 Wave 4 Wave 5 Wave 6  Total
0.2524 0.2881 0.3501 0.0244 0.0276 0.0413 0.9839
```

In Figure 5, the fit of the FMM_{ST} model can be visualized. The goodness of fit of the model is excellent, achieving an $R^2 = 0.9839$.

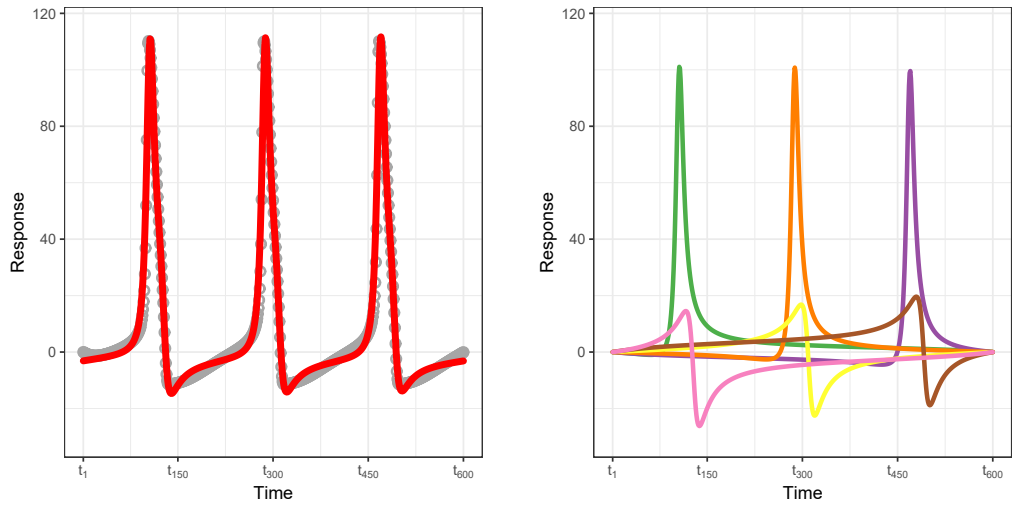


Figure 5: Neuronal APs simulated with the Hodgkin-Huxley model (parameters: $C = 1, g_{Na} = 232, g_K = 45, g_L = 0.215, V_K = -12, V_{Na} = 115, V_L = 10.6, \widetilde{a}_n = 0.95, \widetilde{b}_n = 1.3, \widetilde{a}_m = 1, \widetilde{b}_m = 1.15, \widetilde{a}_h = 1, \widetilde{b}_h = 1$ and applying a short square current of 4.5 μA for 1 millisecond) and the estimated FMM_{ST} signal in red. The components plot of the model can be seen on the right hand side of the figure.

6 Summary

A general overview on the R package **FMM**, which implements the estimation of FMM models, is provided in this paper. The flexibility offered by these models to fit oscillatory signals of many different shapes makes them a very useful tool to model complex rhythmic patterns. The FMM methodology and its application to very diverse biological data has been described in previous papers (Rueda et al., 2019, 2021b,c) and recently revised in Rueda et al. (2021a).

The package allows both single and multicomponent FMM models to be estimated. In order to provide greater flexibility, equality constraints for shape parameters have also been implemented. In addition, graphical representations of the fitted models and the possibility of generating synthetic data are available. The functionality of the package has been illustrated by simulated data and also by real examples from different areas of application related to present-day biological problems. The latest release of the **FMM** package is publicly available on CRAN (http://CRAN.R-project.org/package=FMM). A development version is also provided via GitHub at https://github.com/alexARC26/FMM where code contributions and bugs can be reported.

Possible future extensions of the **FMM** package include the implementation of additional restrictions to suit the model to other real signals; the possibility to include weights that determine how much each observation influences the parameter estimates; and the choice of an optimization technique, other than the Neldel-Mead method, in the estimation algorithm.

7 Acknowledgments

This research was partially supported by the Spanish Ministerio de Economía y Competitividad, grant PID2019-106363RB-I00, as well as the Call for predoctoral contracts of the UVa 2020. The authors thank the editors and two anonymous reviewers for their constructive comments that helped to improve this paper and the described package.

Bibliography

R. C. Anafi, L. J. Francey, J. B. Hogenesch, and J. Kim. CYCLOPS reveals human transcriptional rhythms in health and disease. *Proceedings of the National Academy of Sciences*, 114(20):5312–5317, 2017. URL https://doi.org/10.1073/pnas.1619320114. [p371]

B. Auguie. *gridExtra: Miscellaneous Functions for Grid Graphics*, 2017. URL https://CRAN.R-project.org/package=gridExtra. R package version 2.3. [p370]

A. Bayes de Luna. *Basic Electrocardiography: Normal and Abnormal ECG Patterns*. John Wiley & Sons, Ltd, 2007. URL https://doi.org/10.1002/9780470692622. [p371]

B. Boashash. *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*. Academic Press, San Francisco, CA, 2nd edition, 2016. [p361]

M. Carlucci, A. Krišč iūnas, H. Li, P. Gibas, K. Koncevičius, A. Petronis, and G. Oh. DiscoRhythm: An easy-to-use web application and R package for discovering rhythmicity. *Bioinformatics*, 36(6): 1952–1954, 2020. URL https://doi.org/10.1093/bioinformatics/btz834. [p361]

Caro-Martín, Delgado-García, Gruart, and Sánchez-Campusano. Spike sorting based on shape, phase, and distribution features, and K-TOPS clustering with validity and error indices. *Scientific Reports*, 8 (1):1–28, 2018. URL https://doi.org/10.1038/s41598-018-35491-4. [p373]

W. Chang, J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. *shiny: Web Application Framework for R*, 2021. URL https://CRAN.R-project.org/package=shiny. R package version 1.6.0. [p361]

G. Cornelissen. Cosinor-based rhythmometry. *Theoretical Biology and Medical Modelling*, 11:16, 2014. URL https://doi.org/10.1186/1742-4682-11-16. [p361, 371]

M. Corporation and S. Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2020. URL https://CRAN.R-project.org/package=doParallel. R package version 1.0.16. [p365]

T. D. Downs and K. V. Mardia. Circular regression. *Biometrika*, 89(3):683–698, 2002. URL https://doi.org/10.1093/biomet/89.3.683. [p362]

I. Fernández, A. Rodríguez-Collado, Y. Larriba, A. Lamela, C. Canedo, and C. Rueda. *FMM: Rhythmic Patterns Modeling by FMM Models*, 2021. URL https://CRAN.R-project.org/package=FMM. R package version 0.3.0. [p362]

W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014. URL https://doi.org/10.1017/CBO9781107447615. [p373, 374]

C. L. Gierke, R. Helget, and G. Cornelissen-Guillaume. *CATkit: Chronomics Analysis Toolkit (CAT): Periodicity Analysis*, 2018. URL https://CRAN.R-project.org/package=CATkit. R package version 3.3.3. [p361]

A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):E215–20, 2000. URL https://doi.org/10.1161/01.cir.101.23.e215. [p371]

N. W. Gouwens, S. A. Sorense, J. Berg, C. Lee, et al. Classification of electrophysiological and morphological neuron types in the mouse visual cortex. *Nature Neuroscience*, 22:1182–1195, 2019. URL https://doi.org/10.1038/s41593-019-0417-0. [p373]

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952. URL https://doi.org/10.1113/jphysiol.1952.sp004764. [p373]

M. E. Hughes, J. B. Hogenesch, and K. Kornacker. JTK_CYCLE: An efficient nonparametric algorithm for detecting rhythmic components in genome scale data sets. *Journal of Biological Rhythms*, 25(5): 372–380, 2010. URL https://doi.org/10.1177/0748730410379711. [p361]

S. Kato, K. Shimizu, and G. S. Shieh. A circular - circular regression model. *Statistica Sinica*, 18(2): 633–645, 2008. URL https://www.jstor.org/stable/24308499. [p362]

P. Laguna, R. G. Mark, A. Goldberg, and G. B. Moody. A database for evaluation of algorithms for measurement of qt and other waveform intervals in the ecg. In *Computers in Cardiology 1997*, pages 673–676. IEEE, 1997. URL https://doi.org/10.1109/CIC.1997.648140. [p371]

Y. Larriba, C. Rueda, M. A. Fernández, and S. D. Peddada. A bootstrap based measure robust to the choice of normalization methods for detecting rhythmic features in high dimensional data. *Frontiers in Genetics*, 9:24, 2018. URL https://doi.org/10.3389/fgene.2018.00024. [p371]

Y. Larriba, C. Rueda, M. A. Fernández, and S. D. Peddada. Order restricted inference in chronobiology. *Statistics in Medicine*, 39(3):265–278, 2020. URL https://doi.org/10.1002/sim.8397. [p371]

E. P. Lynch and C. J. Houghton. Parameter estimation of neuron models using in-vitro and in-vivo electrophysiological data. *Frontiers in Neuroinformatics*, 9:10, 2015. URL https://doi.org/10.3389/fninf.2015.00010. [p374]

S. Meek and F. Morris. Introduction. I—Leads, rate, rhythm, and cardiac axis. *BMJ*, 324(7334):415–418, 2002. ISSN 0959-8138. doi: 10.1136/bmj.324.7334.415. [p371]

J. Mermet, J. Yeung, and F. Naef. Systems chronobiology: Global analysis of gene regulation in a 24-hour periodic world. *Cold Spring Harbor Perspectives in Biology*, 9(3), 2017. URL https://doi.org/10.1101/cshperspect.a028720. [p371]

C. P. Mosher, Y. Wei, J. Kamiński, A. Nandi, A. N. Mamelak, C. A. Anastassiou, and U. Rutishauser. Cellular classes in the human brain revealed in vivo by heartbeat-related modulation of the extracellular action potential waveform. *Cell Reports*, 30(10):3536–3551.e6, 2020. URL https://doi.org/10.1016/j.celrep.2020.02.027. [p373]

M. Moskon. CosinorPy: A python package for cosinor-based rhythmometry. *BMC Bioinformatics*, 21 (1):485, 2020. URL https://doi.org/10.1186/s12859-020-03830-w. [p361]

A. Mutak. *cosinor2: Extended Tools for Cosinor Analysis of Rhythms*, 2018. URL https://CRAN.R-project.org/package=cosinor2. R package version 0.2.1. [p361]

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4): 308–313, 1965. URL https://doi.org/10.1093/comjnl/7.4.308. [p363]

E. Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL https://CRAN.R-project.org/package=RColorBrewer. R package version 1.1-2. [p365]

R. Parsons, R. Parsons, N. Garner, H. Oster, and O. Rawashdeh. CircaCompare: A method to estimate and statistically support differences in mesor, amplitude and phase, between circadian rhythms. *Bioinformatics*, 36(4):1208–1212, 2020. URL https://doi.org/10.1093/bioinformatics/btz730. [p361]

M. Rácz, C. Liber, E. Németh, R. Fiáth, J. Rokai, I. Harmati, I. Ulbert, and G. Márton. Spike detection and sorting with deep learning. *Journal of Neural Engineering*, 17(1):016038, 2020. URL https://doi.org/10.1088/1741-2552/ab4896. [p373]

W. Revelle. *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois, 2021. URL https://CRAN.R-project.org/package=psych. R package version 2.1.6. [p361]

A. Rodríguez-Collado and C. Rueda. A simple parametric representation of the Hodgkin-Huxley model. *PLoS ONE*, 16(7):e0254152, 2021a. URL https://doi.org/110.1371/journal.pone.0254152. [p362, 364, 374]

A. Rodríguez-Collado and C. Rueda. Electrophysiological and transcriptomic features reveal a circular taxonomy of cortical neurons. *Frontiers in Human Neuroscience*, 15:410, 2021b. URL https://doi.org/10.3389/fnhum.2021.684950. [p373]

C. Rueda, Y. Larriba, and S. D. Peddada. Frequency modulated Möbius model accurately predicts rhythmic signals in biological and physical sciences. *Scientific Reports*, 9(1):18701, 2019. URL https://doi.org/10.1038/s41598-019-54569-1. [p362, 375]

C. Rueda, I. Fernández, Y. Larriba, and A. Rodríguez-Collado. The FMM approach to analyze biomedical signals: Theory, software, applications and future. *Mathematics*, 9(10):1145, 2021a. URL https://doi.org/10.3390/math9101145. [p375]

C. Rueda, Y. Larriba, and A. Lamela. The hidden waves in the ECG uncovered revealing a sound automated interpretation method. *Scientific Reports*, 11:3724, 2021b. URL https://doi.org/10.1038/s41598-021-82520-w. [p362, 372, 375]

C. Rueda, A. Rodríguez-Collado, and Y. Larriba. A novel wave decomposition for oscillatory signals. *IEEE Transactionns on Signal Processing*, 69:960–972, 2021c. URL https://doi.org/10.1109/TSP.2021.3051428. [p362, 375]

M. Sachs. *cosinor: Tools for estimating and predicting the cosinor model*, 2014. URL https://CRAN.R-project.org/package=cosinor. R package version 1.1. [p361]

A. S. Shah. *card: Cardiovascular and Autonomic Research Design*, 2020. URL https://CRAN.R-project.org/package=card. R package version 0.1.0. [p361]

J. M. Singer and J. J. Hughey. LimoRhyde: A flexible approach for differential analysis of rhythmic transcriptome data. *Journal of Biological Rhythms*, 34(1):5–18, 2019. URL https://doi.org/10.1177/0748730418813785. [p361]

B. C. Souza, V. L. dos Santos, J. Bacelo, and A. B. Tort. Spike sorting with Gaussian mixture models. *Scientific Reports*, 9(1):1–14, 2019. URL https://doi.org/10.1038/s41598-019-39986-6. [p373]

C. Teeter, R. Iyer, V. Menon, N. Gouwens, D. Feng, J. Berg, A. Szafer, N. Cain, H. Zeng, M. Hawrylycz, C. Koch, and S. Mihalas. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature Communications*, 9(1):1–15, 2018. URL https://doi.org/10.1038/s41467-017-02717-4. [p373, 374]

P. F. Thaben and P. O. Westermark. Detecting rhythms in time series with RAIN. *Journal of Biological Rhythms*, 29(6):391–400, 2014. URL https://doi.org/10.1177/0748730414553029. [p361]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p365]

R. Zhang, N. F. Lahens, H. I. Ballance, M. E. Hughes, and J. B. Hogenesch. A circadian gene expression atlas in mammals: Implications for biology and medicine. *Proceedings of the National Academy of Sciences*, 111(45):16219–16224, 2014. URL https://doi.org/10.1073/pnas.1408886111. [p371]

Itziar Fernández
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0002-5077-4448
itziar.fernandez@uva.es

Alejandro Rodríguez-Collado
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0001-5450-9580
alejandro.rodriguez.collado@uva.es

Yolanda Larriba
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0003-0254-4928
yolanda.larriba@uva.es

Adrián Lamela
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0002-7155-8832
adrianlamela@gmail.com

Christian Canedo
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0001-6731-7369
christian.canedo@alumnos.uva.es

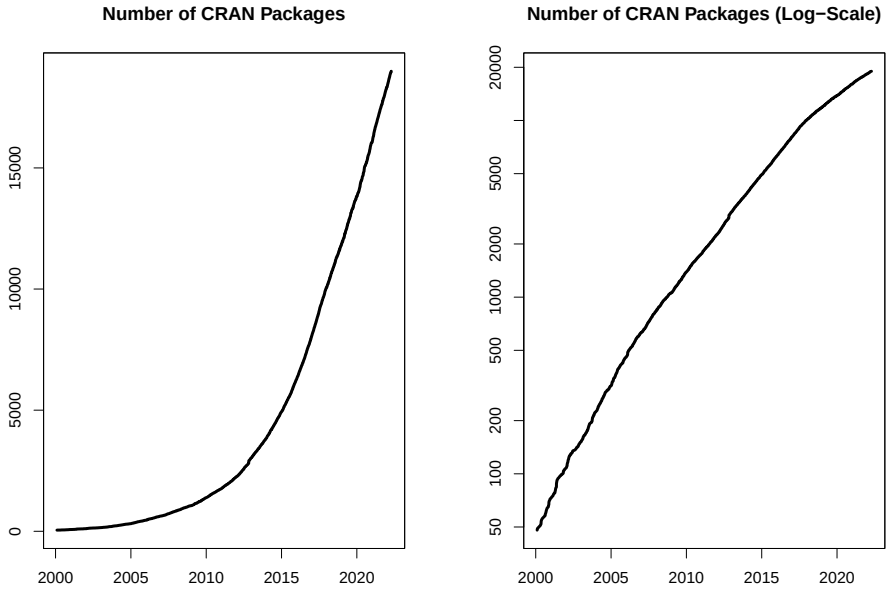
Cristina Rueda
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0001-9638-8991
cristina.rueda@uva.es

Changes on CRAN

2022-01-01 to 2022-03-31

by Kurt Hornik, Uwe Ligges and Achim Zeileis

In the past 3 months, 617 new packages were added to the CRAN package repository. 86 packages were unarchived and 307 were archived. The following shows the growth of the number of active packages in the CRAN package repository:



On 2022-03-31, the number of active packages was around 18924.

Changes in the CRAN Repository Policy

The Policy now says the following:

- (**Using external C/C++/Fortran libraries.**) Where a package wishes to make use of a library not written solely for the package, the package installation should first look to see if it is already installed and if so is of a suitable version. In case not, it is desirable to include the library sources in the package and compile them as part of package installation. If the sources are too large, it is acceptable to download them as part of installation, but do ensure that the download is of a fixed version rather than the latest. Only as a last resort and with the agreement of the CRAN team should a package download pre-compiled software.

  On Windows and macOS static libraries must be used. A separate document, External Libraries for CRAN packages, covers what external libraries are or could be made available.

CRAN package submissions

During the first 4 months of 2022 (January 2022 to April 2022), CRAN received 9601 package submissions. For these, 17170 actions took place of which 11232 (65%) were auto processed actions and 5938 (35%) manual actions.

Minus some special cases, a summary of the auto-processed and manually triggered actions follows:

	archive	inspect	newbies	pending	pretest	publish	recheck	waiting
auto	2391	2716	1392	0	0	3018	1017	698
manual	1893	93	487	323	106	2232	637	167

These include the final decisions for the submissions which were

action	archive	publish
auto	2232 (23.9%)	2479 (26.5%)
manual	1870 (20.0%)	2758 (29.5%)

where we only count those as *auto* processed whose publication or rejection happened automatically in all steps.

A new team member, Viktoria Wimmer, joined the CRAN submission team. Welcome, Viktoria. Unfortunately, Julia Haider left the CRAN submission team after processing 3517 incoming submissions. Thanks a lot!

CRAN mirror security

Currently, there are 102 official CRAN mirrors, 81 of which provide both secure downloads via '`https`' *and* use secure mirroring from the CRAN master (via rsync through ssh tunnels). Since the R 3.4.0 release, `chooseCRANmirror()` offers these mirrors in preference to the others which are not fully secured (yet).

CRAN Task View Initiative

The transition of the established task views to the new workflow on GitHub (https://github.com/cran-task-views/ctv/) that was announced in the previous volume of the journal has been completed (see also https://twitter.com/AchimZeileis/status/1510945091980038145).

Each task view now links to a GitHub repository where it is possible to post issues and make pull requests for proposing improvements – in addition to sending e-mails to the maintainer address which is still possible, of course. Moreover, the task view web pages contain further improvements like a citation, installation notes, and a streamlined overview of core and regular (and currently archived) packages in the task view.

Proposals of new task views are now also possible on GitHub. In fact, a few have already been made for the topics causal inference, genetics and genomics (as a follow-up to the orphaned and archived *Genetics* and *Phylogenetics* views), and sports analytics.

We look forward to further user contributions to the initiative!

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Uwe Ligges
TU Dortmund, Germany
Uwe.Ligges@R-project.org

Achim Zeileis
Universität Innsbruck, Austria
Achim.Zeileis@R-project.org

R Foundation News

by Torsten Hothorn

1 Donations and members

Membership fees and donations received between 2021-12-22 and 2022-03-30.

Donations

RV Detailing Pros of San Diego (United States) Shalese Fitzgerald (United States) Ken Ikeda (Japan) Rees Morrison (United States) Rav Vaid (United States) Clay Valarezo (United States) Kai Wu (China)

Supporting benefactors

Università degli Studi di Padova, Padova (Italy)

Supporting institutions

Institute of Botany of the Czech Academy of Sciences, Pruhonice (Czechia) oikostat GmbH, Ettiswil (Switzerland)

Supporting members

Constantin Ahlmann-Eltze (Germany) Vedo Alagic (Austria) Takaharu Araki (Japan) Frederic Bertrand (France) Michael Blanks (United States) Cédric Chambru (Switzerland) John Chandler (United States) Michael Chirico (United States) Gerard Conaghan (United Kingdom) Brandon Dahl (United States) Michael Dorman (Israel) Fraser Edwards (United Kingdom) Johan Eklund (Sweden) S Ellison (United Kingdom) Dane Evans (United States) Isaac Florence (United Kingdom) Neil Frazer (United States) Bernd Fröhlich (Germany) Jan Marvin Garbuszus (Germany) Gabriel Gersztein (Brazil) Brian Gramberg (Netherlands) Spencer Graves (United States) Hlynur Hallgrímsson (Iceland) Philippe Heymans Smith (Costa Rica) Alexander Huelle (Germany) Heidi Imker (United States) Gavin Kirby (United Kingdom) Ziyad Knio (United States) Gen Kobayashi (Japan) Adrien Le Guillou (France) Yuewei Liu (China) Myriam Maumy (France) Daniel McNichol (United States) Bogdan-Alexandru Micu (Luxembourg) Ernst Molitor (Germany) David Monterde (Spain) Stefan Moog (Germany) Steffen Moritz (Germany) Antonio Paez (Canada) Fergus Reig Gracia (Spain) Stefano Rezzonico (Canada) Ingo Ruczinski (United States) Choonghyun Ryu (Korea, Republic of) Dejan Schuster (Germany) John Smith (United States) Harald Sterly (Germany) Kai Streicher (Switzerland) Robert van den Berg (Austria) Dr. Alfred Wagner (Germany) Petr Waldauf (Czechia) Fredrik Wartenberg (Sweden) 广宇曾(China)

Torsten Hothorn
Universität Zürich, Switzerland Torsten.Hothorn@R-project.org