

The Journal

Volume 14/2, June 2022

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

Contents

Editorial 4

Contributed Research Articles

brolgar: An R package to BRowse Over Longitudinal Data Graphically and Analytically in R	6
The Concordance Test, an Alternative to Kruskal-Wallis Based on the Kendall- τ Distance: An R Package	26
hstestClust: A Package for Marginal Inference of Clustered Data Under Informative Cluster Size	54
akc: A Tidy Framework for Automatic Knowledge Classification in R.	67
APCI: An R and Stata Package for Visualizing and Analyzing Age-Period-Cohort Data	77
shinybrms: Fitting Bayesian Regression Models Using a Graphical User Interface for the R Package brms.	96
Quantifying Population Movement Using a Novel Implementation of Digital Image Correlation in the ICvectorfields Package	121
Refreg: An R Package for Estimating Conditional Reference Regions	135
TensorTest2D: Fitting Generalized Linear Models with Matrix Covariates	152
wavScalogram: An R Package with Wavelet Scalogram Tools for Time Series Analysis.	164
ClusTorus: An R Package for Prediction and Clustering on the Torus by Conformal Prediction	186
kStatistics: Unbiased Estimates of Joint Cumulant Products from the Multivariate Faà Di Bruno's Formula	208
iccCounts: An R Package to Estimate the Intraclass Correlation Coefficient for Assessing Agreement with Count Data	229
R-miss-tastic: a unified platform for missing values methods and workflows.	244
An Open-Source Implementation of the CMPS Algorithm for Assessing Similarity of Bullets	267
rassta: Raster-Based Spatial Stratification Algorithms	286
PDFEstimator: An R Package for Density Estimation and Analysis	305
reclin2: a Toolkit for Record Linkage and Deduplication.	320

News and Notes

News from the Bioconductor Project	329
Changes on CRAN	332
News from the Forwards Taskforce	334
Changes in R	336
R Foundation News	339
The Conference Report of Why R? Turkey 2022: The First R Conference with Call For Papers in Turkey	340

The R Journal is a peer-reviewed publication of the R Foundation for Statistical Computing. Communications regarding this publication should be addressed to the editors. All articles are licensed under the Creative Commons Attribution 4.0 International license (CC BY 4.0, <http://creativecommons.org/licenses/by/4.0/>).

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

Editor-in-Chief:

Catherine Hurley, Maynooth University, Ireland

Executive editors:

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 2 of the R Journal.

First, some news about the journal board. Mark van der Loo has very kindly agreed to move from Associate Editor to Executive Editor to fill a temporary gap. One new Associate Editor, Kevin Burke, has recently joined the team.

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.

There are also exciting new efforts in developing software to convert the legacy papers from latex to Rmarkdown, and hence create an html version to complement the pdf. Abhishek Ulayil, funded by the 2022 Google Summer of Code project, supervised by Heather Turner and Di Cook, with collaboration from Christophe Dervieux and Mitch O'Hara-Wild, has created the R package, **texor**, <https://abhi-1u.github.io/texor/>. It converts the legacy latex style into to the new Rmarkdown template, as would be given with the **rjtools** package. This package will be used to slowly, and steadily convert as many past articles into an html version.

If you are currently only a latex author, the **texor** package will get your paper into an Rmarkdown paper, doing the hard-work of the conversion. This is a good opportunity to get a head start on learning how to make reproducible documents. Reproducible documents keeps your code and results in the same place, and reduces the chance of getting them out of sync. Going forwards with the R Journal there will be a growing emphasis on receiving papers in Rmarkdown (and Quarto, at some point) format, because it is easier to test the code, and it makes the work more accessible to readers.

1 In this issue

News from the CRAN, the R Foundation and the Forwards Taskforce are included in this issue. We also have a report from the Why R? Turkey 2022 conference.

This issue features 18 contributed research articles the majority of which relate to R packages for modelling tasks. All packages are available on CRAN. Topics covered are:

- **Statistical modelling and inference**
 - **APCI**: An R and Stata Package for Age-Period-Cohort Analysis
 - **refreg**: an R package for estimating conditional reference regions
 - From the multivariate Faà di Bruno's formula to unbiased estimates of joint cumulant products: the **kStatistics** package in R
 - The Concordance Test, an Alternative to Kruskal-Wallis Based on the Kendall tau Distance: An R Package
 - **shinybrms**: Fitting Bayesian Regression Models Using a Graphical User Interface for the R Package brms
 - **PDFEstimator**: An R Package for Density Estimation and Analysis
 - **htestClust**: Hypothesis Tests for Clustered Data under Informative Cluster Size in R
 - **ClusTorus**: An R Package for Prediction and Clustering on the Torus by Conformal Prediction
 - **TensorTest2D**: Fitting Generalized Linear Models with Matrix Covariates
- **Ecological and Environmental analysis**
 - Quantifying Population Movement Using a Novel Implementation of Digital Image Correlation in the **ICvectorfields** package

- **iccCounts**: an R Package to Estimate the Intraclass Correlation Coefficient for Assessing Agreement with Count Data
- **rassta**: Raster-based Spatial Stratification Algorithms
- **Missing data**
 - R-miss-tastic: a unified platform for missing values methods and workflows
 - **reclin2**: a Toolkit for Record Linkage and Deduplication
- **Time Series Analysis**
 - **wavScalogram**: an R package with wavelet scalogram tools for time series analysis
 - **brolgar**: An R package to BRowse Over Longitudinal Data Graphically and Analytically in R”
- **Other**
 - **akc**: A tidy framework for automatic knowledge classification in R
 - An Open-Source Implementation of the CMPS Algorithm for Assessing Similarity of Bullets

Catherine Hurley
Maynooth University

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

brolgar: An R package to BRowse Over Longitudinal Data Graphically and Analytically in R

by Nicholas Tierney, Dianne Cook, and Tania Provan

Abstract Longitudinal (panel) data provide the opportunity to examine temporal patterns of individuals, because measurements are collected on the same person at different, and often irregular, time points. The data is typically visualised using a “spaghetti plot”, where a line plot is drawn for each individual. When overlaid in one plot, it can have the appearance of a bowl of spaghetti. With even a small number of subjects, these plots are too overloaded to be read easily. The interesting aspects of individual differences are lost in the noise. Longitudinal data is often modelled with a hierarchical linear model to capture the overall trends, and variation among individuals, while accounting for various levels of dependence. However, these models can be difficult to fit, and can miss unusual individual patterns. Better visual tools can help to diagnose longitudinal models, and better capture the individual experiences. This paper introduces the R package, **brolgar** (BRowse over Longitudinal data Graphically and Analytically in R), which provides tools to identify and summarise interesting individual patterns in longitudinal data.

1 Introduction

This paper is about exploring longitudinal data effectively. By “longitudinal data” we specifically mean individuals repeatedly measured through time. This could include panel data, where possibly different samples from a key variable (e.g. country), are aggregated at each time collection. The important component is a key variable with repeated measurements regularly, or irregularly over time. The inherent structure allows us to examine temporal patterns of individuals, shown in Figure 1, of the average height of Australian males over years. The individual component is country, and the time component is year. The variable country along with other variables is measured repeatedly from 1900 to 1970, with irregular intervals between years.

The full dataset of Figure 1 is shown in Figure 2, showing 144 countries from the year 1700. This plot is challenging to understand because there is overplotting, making it hard to see the individuals. Solutions to this are not always obvious. Showing separate individual plots of each country does not help, as 144 plots is too many to comprehend. Making the lines transparent or fitting a simple model to all the data Figure 2B, might be a common first step to see common trends. However, all this seems to clarify is: 1) There is a set of some countries that are similar, and they are distributed around the center of the countries, and 2) there is a general upward trend in heights over time. We learn about the collective, but lose sight of the individuals.

This paper demonstrates how to effectively and efficiently explore longitudinal data, using the R package, **brolgar**. We examine four problems in exploring longitudinal data:

1. How to sample the data
2. Finding interesting individuals
3. Finding representative individuals

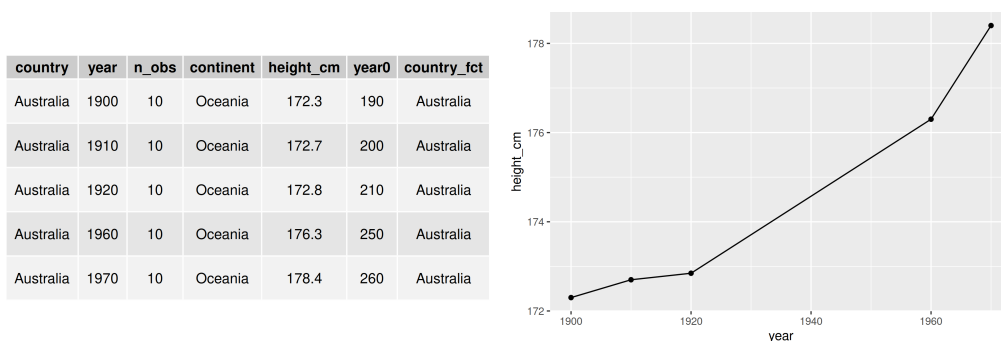


Figure 1: Example of longitudinal data: average height of men in Australia for 1900-1970. The height increase over time, and are measured at irregular intervals.

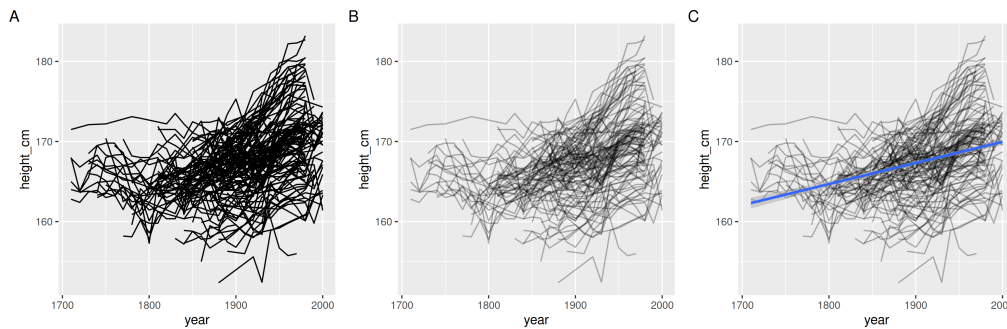


Figure 2: The full dataset shown as a spaghetti plot (A), with transparency (B), and with a linear model overlaid (C). It is still hard to see the individuals.

4. Understanding a model

This paper proceeds in the following way: first, a brief review of existing approaches to longitudinal data, then the definition of longitudinal data, then approaches to these four problems are discussed, followed by a summary.

2 Background

R provides basic time series, `ts`, objects, which are vectors or matrices that represent data sampled at equally spaced points in time. These have been extended through packages such as `xts`, and `zoo` (Ryan and Ulrich 2020; Zeileis and Grothendieck 2005), which only consider data in a wide format with a regular implied time series. These are not appropriate for longitudinal data, which can have indexes that are not time unit oriented, such as “Wave 1...n”, or may contain irregular intervals.

Other packages focus more directly on panel data in R, focussing on data operations and model interfaces. The `pmdplyr` package provides “Panel Manoeuvres” in `dplyr` (Huntington-Klein and Khor 2020). It defines the data structure in as a `pibble` object (`panel tibble`), requiring an `id` and group column being defined to identify the unique identifier and grouping. The `pmdplyr` package focuses on efficient and custom joins and functions, such as `inexact_left_join()`. It does not implement tidyverse equivalent tools, but instead extends their usecase with a new function, for example `mutate_cascade` and `mutate_subset`. The `panelr` package provides an interface for data reshaping on panel data, providing widening and lengthening functions (`widen_panel()` and `long_panel()` (Long 2020)). It also provides model facilitating functions by providing its own interface for mixed effects models. The `plm` package (Millo 2017) for panel data econometrics provides methods for estimating models such as GMM for panel data, and testing, for example for model specification or serial correlation. It also provides a data structure, the `pdata.frame`, which stores the index attribute of the individual and time dimensions, for use within the package’s functions.

These software generally re-implement their own custom panel data class object, as well as custom data cleaning tasks, such as reshaping into long and wide form. They all share similar features, providing some identifying or index variable, and some grouping or key.

3 Longitudinal Data Structures

Longitudinal data is a sibling of many other temporal data forms, including panel data, repeated measures, and time series. The differences are many, and can be in data collection, context and even the field of research. Time series are usually long and regularly spaced in time. Panel data may measure different units at each time point and aggregate these values by a categorical or key variable. Repeated measures typically measure before and after treatment effects. We like to think of longitudinal as measuring the same individual (e.g. wage earner) over time, but this definition is not universally agreed on. Despite the differences, they all share a fundamental similarity: they are measurements over a time period.

This time period has structure - the time component (dates, times, waves, seconds, etc), and the spacing between measurements - unequal or equal. This data structure needs to be respected during analysis to preserve the lowest level of granularity, to avoid for example, collapsing across month when the data is collected every second, or assuming measurements occur at fixed time intervals. These mistakes can be avoided by encoding the data structure into the data itself. This information can then be accessed by analysis tools, providing a consistent way to understand and summarise the

data. This ensures the different types of longitudinal data previously mentioned can be handled in the same way.

Building on a tsibble

Since longitudinal data can be thought of as “individuals repeatedly measured through time”, they can be considered as a type of time series, as defined in Hyndman and Athanasopoulos (2018): “Anything that is observed sequentially over time is a **time series**”. This definition has been realised as a time series `tsibble` in (Wang, Cook, and Hyndman 2020). These objects are defined as data meeting these conditions:

1. The index: the time variable
2. The key: variable(s) defining individual groups (or series)
3. The index and key (1 + 2) together determine a distinct row

If the specified key and index pair do not define a distinct row - for example, if there are duplicates in the data, the `tsibble` will not be created. This helps ensure the data is properly understood and cleaned before analysis is conducted, removing avoidable errors that might have impacted downstream decisions.

We can formally define our heights data from Figure 1 as a `tsibble` using, `as_tsibble`:

```
heights_brolgar <- as_tsibble(heights_brolgar,
                             index = year,
                             key = country,
                             regular = FALSE)
```

The index is `year`, the key is `country`, and `regular = FALSE` since the intervals in the years measured are not regular. Using a `tsibble` means that the index and key time series information is recorded only **once**, and can be referred to many times in other parts of the data analysis by time-aware tools.

In addition to providing consistent ways to manipulate time series data, further benefits to building on `tsibble` are how it works within the tidyverse ecosystem, as well as the tidy time series packages called “tidyverts”, containing `fable` (O’Hara-Wild, Hyndman, and Wang 2020a), `feasts`, (O’Hara-Wild, Hyndman, and Wang 2020b). For example, `tsibble` provides modified tidyverse functions to explore implicit missing values in the index (e.g., `has_gaps()` and `fill_gaps()`), as well as grouping and partitioning based on the index with `index_by()`. For full details and examples of use with the tidyverts time series packages, see Wang, Cook, and Hyndman (2020).

The `brolgar` package uses `tsibble` so users can take advantage of these tools, learning one way of operating a data analysis that will work and have overlap with other contexts.

Characterising Individual Series

Calculating a feature

We can summarise the individual series by collapsing their many measurements into a single statistic, such as the minimum, maximum, or median, with one row per key. We do this with the `features` function from the `fabletools` package, made available in `brolgar`. This provides a summary of a given variable, accounting for the time series structure, and returning one row per key specified. It can be thought of as a time-series aware variant of the `summarise` function from `dplyr`. The feature function works by specifying the data, the variable to summarise, and the feature to calculate. A template is shown below

```
features(<DATA>, <VARIABLE>, <FEATURE>)
```

or, with the pipe:

```
<DATA> %>% features(<VARIABLE>, <FEATURE>)
```

For example, to calculate the minimum height for each key (`country`), in `heights`, we specify the `heights` data, then the variable to calculate features on, `height_cm`, then the feature to calculate, `min` (we write `c(min = min)` so the column calculated gets the name “min”):

```
heights_min <- features(.tbl = heights_brolgar,
  .var = height_cm,
  features = c(min = min))
```

```
heights_min
```

```
#> # A tibble: 119 x 2
#>   country      min
#>   <chr>      <dbl>
#> 1 Afghanistan 161.
#> 2 Algeria      166.
#> 3 Angola       159.
#> 4 Argentina   167.
#> 5 Armenia     164.
#> 6 Australia   170.
#> 7 Austria     162.
#> 8 Azerbaijan  170.
#> 9 Bangladesh  160.
#> 10 Belgium    163.
#> # ... with 109 more rows
```

We call these summaries features of the data. We can use this information to summarise these features of the data, for example, visualising the distribution of minimum values (Figure 3A).

We are not limited to one feature at a time, many features can also be calculated, for example:

```
heights_three <- heights_brolgar %>%
  features(height_cm, c(
    min = min,
    median = median,
    max = max
  ))
```

```
heights_three
```

```
#> # A tibble: 119 x 4
#>   country      min median  max
#>   <chr>      <dbl> <dbl> <dbl>
#> 1 Afghanistan 161.  167. 168.
#> 2 Algeria      166.  169. 171.
#> 3 Angola       159.  167. 169.
#> 4 Argentina   167.  168. 174.
#> 5 Armenia     164.  169. 172.
#> 6 Australia   170.  172. 178.
#> 7 Austria     162.  167. 179.
#> 8 Azerbaijan  170.  172. 172.
#> 9 Bangladesh  160.  162. 164.
#> 10 Belgium    163.  166. 177.
#> # ... with 109 more rows
```

These can then be visualised together (Figure 3).

These sets of features can be pre-specified, for example, `brolgar` provides a five number summary (minimum, 25th quantile, median, mean, 75th quantile, and maximum) of the data with `feat_five_num`:

```
heights_five <- heights_brolgar %>%
  features(height_cm, feat_five_num)
```

```
heights_five
```

```
#> # A tibble: 119 x 6
#>   country      min  q25  med  q75  max
#>   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 Afghanistan 161.  164. 167. 168. 168.
```

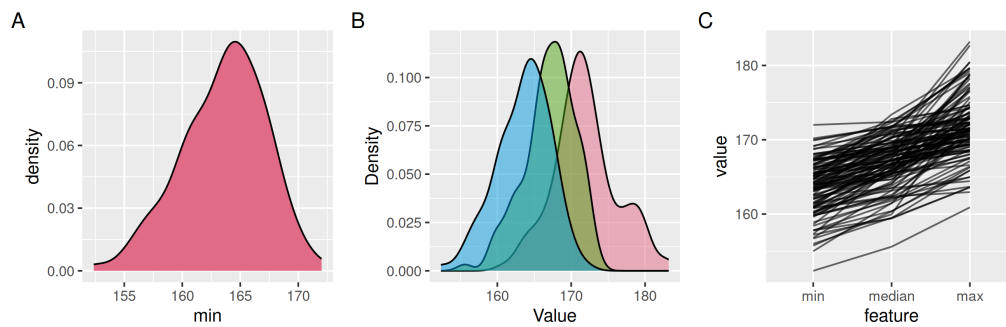


Figure 3: Three plots showing the distribution of minimum, median, and maximum values of height in centimeters. Part A shows just the distribution of minimum, part B shows the distribution of minimum, median, and maximum, and part C shows these three values plotted together as a line graph. We see that there is overlap amongst all three statistics. That is, some countries minimum heights are taller than some countries maximum heights.

```
#> 2 Algeria      166. 168. 169 170. 171.
#> 3 Angola       159. 160. 167. 168. 169.
#> 4 Argentina   167. 168. 168. 170. 174.
#> 5 Armenia     164. 166. 169. 172. 172.
#> 6 Australia   170. 171. 172. 173. 178.
#> 7 Austria     162. 164. 167. 169. 179.
#> 8 Azerbaijan  170. 171. 172. 172. 172.
#> 9 Bangladesh  160. 162. 162. 163. 164.
#> 10 Belgium    163. 164. 166. 168. 177.
#> # ... with 109 more rows
```

This takes the heights data, pipes it to features, and then instructs it to summarise the height_cm variable, using feat_five_num. There are several handy functions for calculating features of the data that brolgar provides. These all start with feat_, and include:

- feat_ranges(): min, max, range difference, interquartile range;
- feat_spread(): variance, standard deviation, median absolute distance, and interquartile range;
- feat_monotonic(): is it always increasing, decreasing, or unvarying?;
- feat_diff_summary(): the summary statistics of the differences amongst a value, including the five number summary, as well as the standard deviation and variance;
- feat_brolgar(), which will calculate all features available in the **brolgar** package.
- Other examples of features from the **feasts** package.

Feature sets

If you want to run many or all features from a package on your data you can collect them all with feature_set. For example:

```
library(fabletools)
feat_set_brolgar <- feature_set(pkgs = "brolgar")
length(feat_set_brolgar)

#> [1] 6
```

You could then run these like so:

```
heights_brolgar %>%
  features(height_cm, feat_set_brolgar)

#> # A tibble: 119 x 46
#>   country      min...1 med...2 max...3 min...4 q25...5 med...6 q75...7 max...8
#>   <chr>         <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 Afghanistan  161.  167.  168.  161.  164.  167.  168.  168.
#> 2 Algeria      166.  169.  171.  166.  168.  169.  170.  171.
#> 3 Angola       159.  167.  169.  159.  160.  167.  168.  169.
```



```
#> 4 Argentina      167.  168.  174.  167.  168.  168.  170.  174.
#> 5 Armenia        164.  169.  172.  164.  166.  169.  172.  172.
#> 6 Australia      170  172.  178.  170  171.  172.  173.  178.
#> 7 Austria        162.  167.  179.  162.  164.  167.  169.  179.
#> 8 Azerbaijan     170.  172.  172.  170.  171.  172.  172.  172.
#> 9 Bangladesh    160.  162.  164.  160.  162.  162.  163.  164.
#> 10 Belgium       163.  166.  177.  163.  164.  166.  168.  177.
#> # ... with 109 more rows, and 37 more variables: min...9 <dbl>, max...10 <dbl>,
#> #   range_diff...11 <dbl>, iqr...12 <dbl>, var...13 <dbl>, sd...14 <dbl>,
#> #   mad...15 <dbl>, iqr...16 <dbl>, min...17 <dbl>, max...18 <dbl>,
#> #   median <dbl>, mean <dbl>, q25...21 <dbl>, q75...22 <dbl>, range1 <dbl>,
#> #   range2 <dbl>, range_diff...25 <dbl>, sd...26 <dbl>, var...27 <dbl>,
#> #   mad...28 <dbl>, iqr...29 <dbl>, increase...30 <dbl>, decrease...31 <dbl>,
#> #   unvary...32 <dbl>, diff_min <dbl>, diff_q25 <dbl>, diff_median <dbl>, ...
```

To see other features available in the `feasts` R package run `library(feasts) then ?fabletools::feature_set`.

Creating your own feature

To create your own features or summaries to pass to features, you provide a named vector of functions. These can include functions that you have written yourself. For example, returning the first three elements of a series, by writing our own second and third functions.

```
second <- function(x) nth(x, n = 2)
third <- function(x) nth(x, n = 3)

feat_first_three <- c(first = first,
                      second = second,
                      third = third)
```

These are then passed to features like so:

```
heights_brolgar %>%
  features(height_cm, feat_first_three)

#> # A tibble: 119 x 4
#>   country      first second third
#>   <chr>      <dbl> <dbl> <dbl>
#> 1 Afghanistan 168.  166.  167.
#> 2 Algeria     169.  166.  169
#> 3 Angola      160.  159.  160.
#> 4 Argentina   170.  168.  168
#> 5 Armenia     169.  168.  166.
#> 6 Australia   170  171.  170.
#> 7 Austria     165.  163.  162.
#> 8 Azerbaijan  170.  171.  171.
#> 9 Bangladesh  162.  162.  164.
#> 10 Belgium    163.  164.  164
#> # ... with 109 more rows
```

As well, `brolgar` provides some useful additional features for the five number summary, `feat_five_num`, whether keys are monotonically increasing `feat_monotonic`, and measures of spread or variation, `feat_spread`. Inside `brolgar`, the features are created with the following syntax:

```
feat_five_num <- function(x, ...) {
  c(
    min = b_min(x, ...),
    q25 = b_q25(x, ...),
    med = b_median(x, ...),
    q75 = b_q75(x, ...),
    max = b_max(x, ...)
  )
}
```

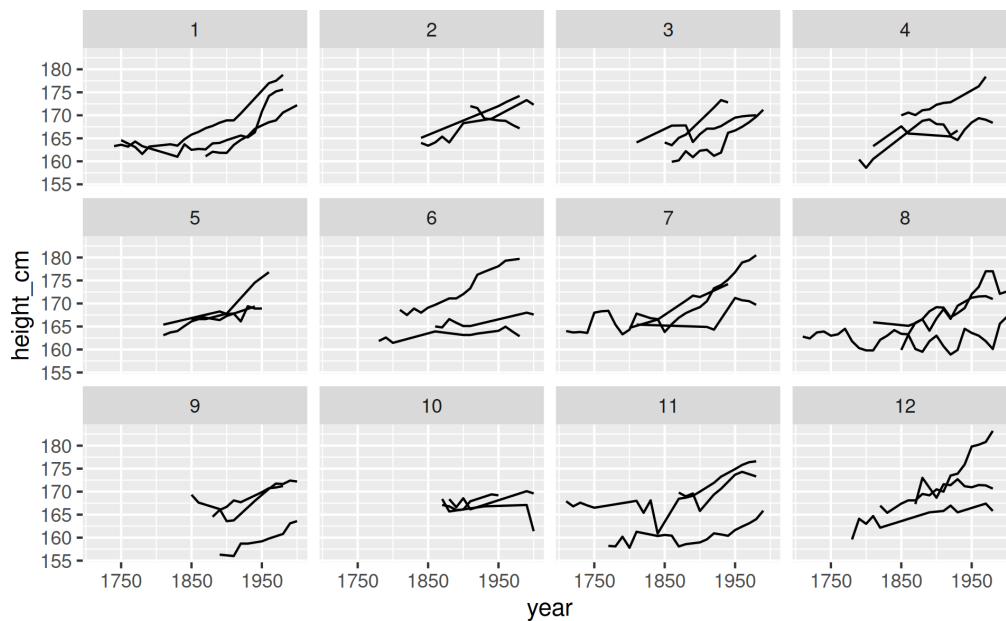


Figure 4: Twelve facets with three keys per facet shown. This allows us to quickly view a random sample of the data.

Here the functions `b_` are functions with a default of `na.rm = TRUE`, and in the cases of quantiles, they use `type = 8`, and `names = FALSE`. What is particularly useful is that these will work on any type of time series data, and you can use other more typical time series features from the `feasts` package, such as autocorrelation, `feat_acf()` and Seasonal and Trend decomposition using Loess `feat_stl()` (O’Hara-Wild, Hyndman, and Wang 2020b).

This demonstrates a workflow that can be used to understand and explore your longitudinal data. The `brolgar` package builds upon this workflow made available by `feasts` and `fabletools`. Users can also create their own features to summarise the data.

4 Breaking up the Spaghetti

Plots like Figure 2 are often called, “spaghetti plots”, and can be useful for a high level understanding as a whole. However, we cannot process and understand the individuals when the data is presented like this.

Sampling

Just how spaghetti is portioned out for consumption, we can sample some of the data by randomly sampling the data into sub-plots with the `facet_sample()` function (Figure 4).

```
ggplot(heights_brolgar,
  aes(x = year,
    y = height_cm,
    group = country)) +
  geom_line() +
  facet_sample() +
  scale_x_continuous(breaks = c(1750, 1850, 1950))
```

This defaults to 12 facets and 3 samples per facet, and provides options for the number of facets, and the number of samples per facet. This means the user only needs to consider the most relevant questions: “How many keys per facet?” and “How many facets to look at?”. The code to change the figure from Figure 2 into 4 requires only one line of code, shown below:

```
ggplot(heights_brolgar,
  aes(x = year,
    y = height_cm,
```

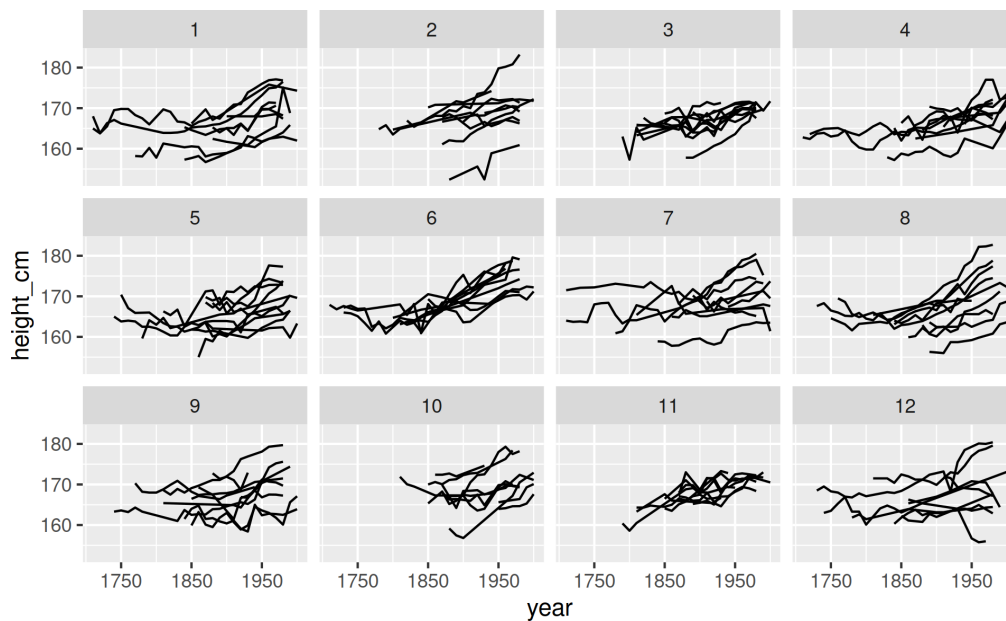


Figure 5: All of the data is shown by spreading out each key across twelve facets. Each key is only shown once, and is randomly allocated to a facet.

```

    group = country)) +
  geom_line() +
  facet_sample()

```

Stratifying

Extending this idea of samples, we can instead look at **all** of the data, spread out equally over facets, using `facet_strata()`. It uses 12 facets by default, controllable with `n_strata`. The code to do so is shown below, creating Figure 5.

```

ggplot(heights_brolgar,
  aes(x = year,
    y = height_cm,
    group = country)) +
  geom_line() +
  facet_strata() +
  scale_x_continuous(breaks = c(1750, 1850, 1950))

```

Featuring

Figure 4 and Figure 5 only show each key once, being randomly assigned to a facet. We can meaningfully place the keys into facets, by arranging the heights “along” a variable, like year, using the `along` argument in `facet_strata` to produce Figure 6:

```

ggplot(heights_brolgar,
  aes(x = year,
    y = height_cm,
    group = country)) +
  geom_line() +
  facet_strata(along = -year) +
  scale_x_continuous(breaks = c(1750, 1850, 1950))

```

We have not lost any of the data, only the order in which they are presented has changed. We learn the distribution and changes in heights over time, and those measured from the earliest times appear to be more similar, but there is much wider variation in the middle years, and then for more recent heights measured from the early 1900s, the heights are more similar. The starting point of each

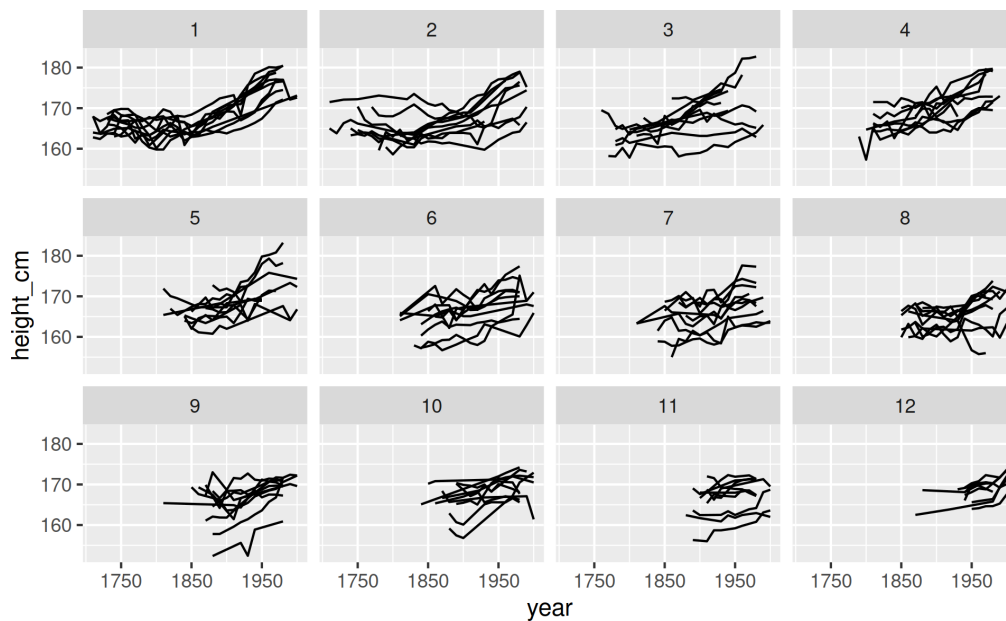


Figure 6: Displaying all the data across twelve facets. Instead of each key being randomly in a facet, each facet displays a specified range of values of year. In this case, the top left facet shows the keys with the earliest starting year, and the bottom right shows the facet with the latest starting year.

of these years seems to increase at roughly the same interval. This informs us that the starting times of the years is approximately uniform.

Together `facet_sample()` and `facet_strata()` allow for rapid exploration, by focusing on relevant questions instead of the minutiae. This is achieved by appropriately randomly assigning while maintaining key structure, keeping the correct number of keys per plot, and so on. For example, `facet_sample()` the questions are: “How many lines per facet” and “How many facets?”, and for `facet_strata()` the questions are: “How many facets / strata?” and “What to arrange plots along?”.

Answering these questions keeps the analysis in line with the analytic goals of exploring the data, rather than distracting to minutiae. This is a key theme of improving tools for data analysis. Abstracting away the parts that are not needed, so the analyst can focus on the task at hand.

Under the hood, `facet_sample()` and `facet_strata()` are powered with `sample_n_keys()` and `stratify_keys()`. These can be used to create data structures used in `facet_sample()` and `facet_strata()`, and extend them for other purposes.

Using a `tsibble` stores important key and index components, in turn allowing for better ways to break up spaghetti plots so we can look at many and all sub-samples using `facet_sample()` and `facet_strata()`.

5 Book-keeping

Longitudinal data is not always measured at the same time and at the same frequency. When exploring longitudinal data, a useful first step is to explore the frequency of measurements of the index. We can check if the index is regular using `index_regular()` and summarise the spacing of the index with `index_summary()`. These are S3 methods, so for `data.frame` objects, the index must be specified, however for the `tsibble` objects, the defined index is used.

```
index_summary(heights_brolgar)
#>   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  1710  1782   1855   1855  1928   2000

index_regular(heights_brolgar)
#> [1] TRUE
```

We can explore how many observations per country by counting the number of observations with features, like so:

```
heights_brolgar %>% features(year, n_obs)
```

```
#> # A tibble: 119 x 2
#>   country      n_obs
#>   <chr>      <int>
#> 1 Afghanistan     5
#> 2 Algeria         5
#> 3 Angola          9
#> 4 Argentina      20
#> 5 Armenia        11
#> 6 Australia      10
#> 7 Austria        18
#> 8 Azerbaijan     7
#> 9 Bangladesh     9
#> 10 Belgium       10
#> # ... with 109 more rows
```

This can be further summarised by counting the number of times there are a given number of observations:

```
heights_brolgar %>% features(year, n_obs) %>% count(n_obs)
```

```
#> # A tibble: 24 x 2
#>   n_obs     n
#>   <int> <int>
#> 1     5    11
#> 2     6    11
#> 3     7    13
#> 4     8     5
#> 5     9    12
#> 6    10    12
#> 7    11     9
#> 8    12     4
#> 9    13     7
#> 10   14     6
#> # ... with 14 more rows
```

Because we are exploring the temporal patterns, we cannot reliably say anything about those individuals with few measurements. The data used, `heights_brolgar` has less than 5 measurements. This was done using `add_n_obs()`, which adds the number of observations to the existing data. Overall this drops 25 countries, leaves us with 119 out of the original 144 countries.

```
heights_brolgar <- heights %>%
  add_n_obs() %>%
  filter(n_obs >= 5)
```

We can further explore when countries are first being measured using `features` to find the first year for each country number of starting years with the `first` function from `dplyr`, and explore this with a visualisation (Figure 7).

```
heights_brolgar %>%
  features(year, c(first = first))
```

```
#> # A tibble: 119 x 2
#>   country      first
#>   <chr>      <dbl>
#> 1 Afghanistan 1870
#> 2 Algeria     1910
#> 3 Angola      1790
#> 4 Argentina   1770
#> 5 Armenia     1850
#> 6 Australia   1850
#> 7 Austria     1750
```

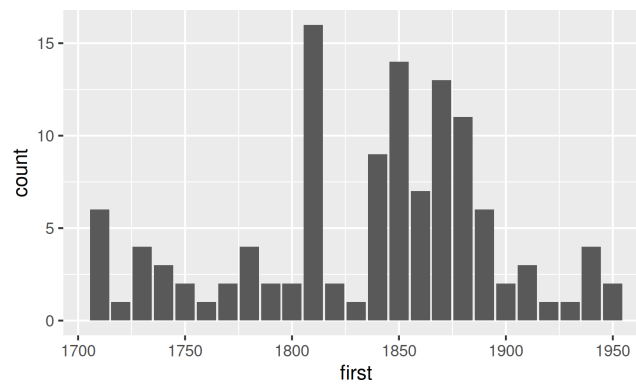


Figure 7: Distribution of starting years of measurement. The data is already binned into 10 year blocks. Most of the years start between 1840 and 1900.

```
#> 8 Azerbaijan 1850
#> 9 Bangladesh 1850
#> 10 Belgium 1810
#> # ... with 109 more rows
```

```
heights_brolgar %>%
  features(year, c(first = first)) %>%
  ggplot(aes(x = first)) +
  geom_bar()
```

We can explore the variation in first year using `feat_diff_summary`. This combines many summaries of the differences in year.

```
heights_diffs <- heights_brolgar %>%
  features(year, feat_diff_summary)
```

```
heights_diffs
```

```
#> # A tibble: 119 x 10
#>   country    diff_min diff_q25 diff_~1 diff_~2 diff_~3 diff_~4 diff_~5 diff_sd
#>   <chr>      <dbl>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
#> 1 Afghanistan 10      10      30    32.5  55.8    60    692.   26.3
#> 2 Algeria     10      10      10    22.5  39.2    60    625    25
#> 3 Angola      10      10      10    17.5  10      70    450    21.2
#> 4 Argentina  10      10      10    11.6  10      40    47.4   6.88
#> 5 Armenia     10      10      10    15    20.8    30    72.2   8.50
#> 6 Australia  10      10      10    13.3  10      40    100    10
#> 7 Austria     10      10      10    13.5  10      40    74.3   8.62
#> 8 Azerbaijan 10      10      10    25    25.8    90   1030   32.1
#> 9 Bangladesh 10      10      10    18.8  15.8    70   441.   21.0
#> 10 Belgium    10      10      10    16.7  23.3    40   125    11.2
#> # ... with 109 more rows, 1 more variable: diff_iqr <dbl>, and abbreviated
#> #   variable names 1: diff_median, 2: diff_mean, 3: diff_q75, 4: diff_max,
#> #   5: diff_var
```

This is particularly useful as using `diff` on year would return a very wide dataset that is hard to explore:

```
heights_brolgar %>%
  features(year, diff)
```

```
#> # A tibble: 119 x 30
#>   country    ...1 ...2 ...3 ...4 ...5 ...6 ...7 ...8 ...9 ...10 ...11
#>   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 Afghanistan 10    50    60    10    NA    NA    NA    NA    NA    NA    NA
#> 2 Algeria     10    10    60    10    NA    NA    NA    NA    NA    NA    NA
```

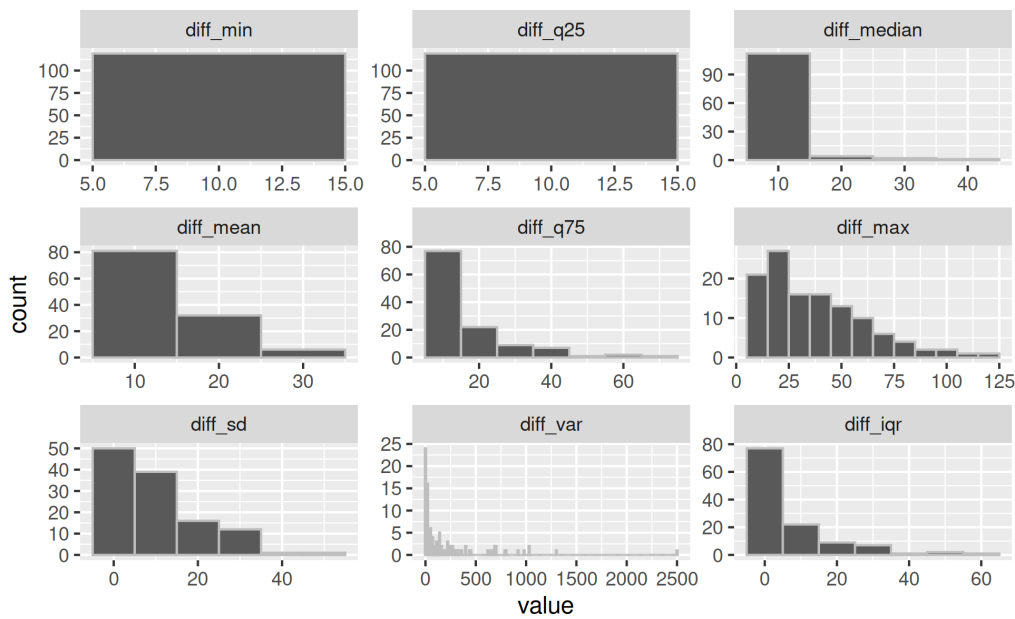


Figure 8: Exploring the different summary statistics of the differences amongst the years. We learn that the smallest interval between measurements is 10 years, and the largest interval is between 10 and 125 years, and that most of the data is measured between 10 and 30 or so years.

```
#> 3 Angola          10  10  70  10  10  10  10  10  NA  NA  NA
#> 4 Argentina      10  10  10  10  10  10  10  10  10  10  10
#> 5 Armenia        10  30  10  10  30  20  10  10  10  10  NA
#> 6 Australia      10  10  10  10  10  10  10  40  10  NA  NA
#> 7 Austria        20  10  10  30  10  10  10  10  10  10  10
#> 8 Azerbaijan     10  90  10  10  10  20  NA  NA  NA  NA  NA
#> 9 Bangladesh     10  10  10  70  10  20  10  10  NA  NA  NA
#> 10 Belgium       10  10  10  10  10  10  30  40  20  NA  NA
#> # ... with 109 more rows, and 18 more variables: ...12 <dbl>, ...13 <dbl>,
#> #   ...14 <dbl>, ...15 <dbl>, ...16 <dbl>, ...17 <dbl>, ...18 <dbl>,
#> #   ...19 <dbl>, ...20 <dbl>, ...21 <dbl>, ...22 <dbl>, ...23 <dbl>,
#> #   ...24 <dbl>, ...25 <dbl>, ...26 <dbl>, ...27 <dbl>, ...28 <dbl>,
#> #   ...29 <dbl>
```

We can then look at the summaries of the differences in year by changing to long form and facetting (Figure 8), we learn about the range of intervals between measurements, the smallest being 10 years, the largest being 125, and that most of the data is measured between 10 and 30 years.

6 Finding Waldo

Looking at a spaghetti plot, it can be hard to identify which lines are the most interesting, or unusual. A workflow to identify interesting individuals to start with is given below:

1. Decide upon an interesting feature (e.g., maximum)
2. This feature produces one value per key
3. Examine the distribution of the feature
4. Join this table back to the data to get all observations for those keys
5. Arrange the keys or filter, using the feature
6. Display the data for selected keys

This workflow is now demonstrated. Firstly, we **decide on an interesting feature**, “maximum height”, and whether height is always increasing. We calculate our own “feature”, calculating maximum height, and whether a value is increasing (with `brolgar`’s `increasing` function) as follows:

```
heights_max_in <- heights_brolgar %>%
  features(height_cm, list(max = max,
                           increase = increasing))
```

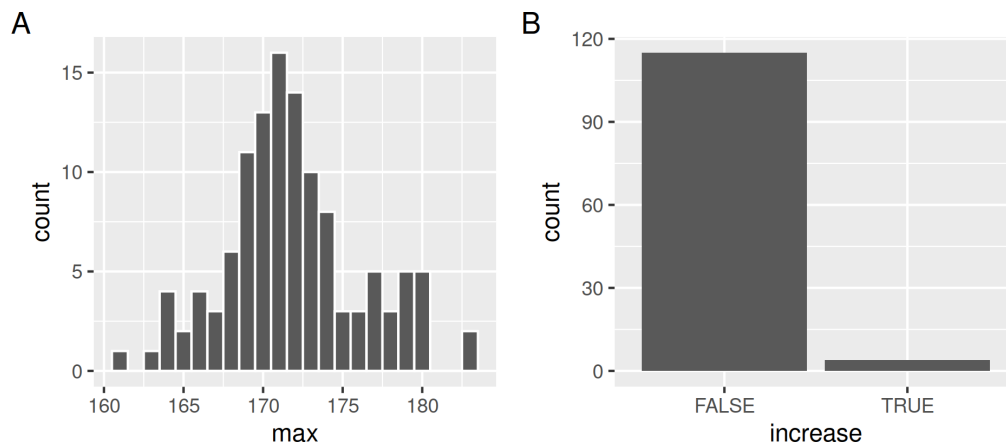


Figure 9: The different distributions of the features - A is depicting the distribution of maximum height, and B displays the number of countries that are always increasing (FALSE), and always increasing (TRUE). We note that the average maximum heights range from about 160cm to 185cm, with most being around 170cm. We also learn that the vast majority of countries are not always increasing in height through time.

```
heights_max_in
```

```
#> # A tibble: 119 x 3
#>   country      max increase
#>   <chr>      <dbl> <lgl>
#> 1 Afghanistan 168. FALSE
#> 2 Algeria     171. FALSE
#> 3 Angola      169. FALSE
#> 4 Argentina   174. FALSE
#> 5 Armenia     172. FALSE
#> 6 Australia   178. FALSE
#> 7 Austria     179. FALSE
#> 8 Azerbaijan  172. FALSE
#> 9 Bangladesh  164. FALSE
#> 10 Belgium    177. FALSE
#> # ... with 109 more rows
```

This returns a dataset of **one value per key**. Figure 9 examines the distribution of the features, showing us the distribution of maximum height, and the number of countries that are always increasing.

We can now **join this table back to the data to get all observations for those keys** to move from one key per row to all many rows per key.

```
heights_max_in_full <- heights_max_in %>%
  left_join(heights_brolgar,
            by = "country")
```

```
heights_max_in_full
```

```
#> # A tibble: 1,406 x 9
#>   country      max increase  year n_obs continent height_cm year0 country_fct
#>   <chr>      <dbl> <lgl>  <dbl> <int> <chr>      <dbl> <dbl> <fct>
#> 1 Afghanistan 168. FALSE  1870     5 Asia      168.  160 Afghanistan
#> 2 Afghanistan 168. FALSE  1880     5 Asia      166.  170 Afghanistan
#> 3 Afghanistan 168. FALSE  1930     5 Asia      167.  220 Afghanistan
#> 4 Afghanistan 168. FALSE  1990     5 Asia      167.  280 Afghanistan
#> 5 Afghanistan 168. FALSE  2000     5 Asia      161.  290 Afghanistan
#> 6 Algeria     171. FALSE  1910     5 Africa   169.  200 Algeria
#> 7 Algeria     171. FALSE  1920     5 Africa   166.  210 Algeria
#> 8 Algeria     171. FALSE  1930     5 Africa   169.  220 Algeria
```



```
#> 9 Algeria      171. FALSE    1990    5 Africa      171.    280 Algeria
#> 10 Algeria     171. FALSE    2000    5 Africa      170.    290 Algeria
#> # ... with 1,396 more rows
```

We can then **arrange the keys or filter, using the feature**, for example, filtering only those countries that are only increasing:

```
heights_increase <- heights_max_in_full %>% filter(increase)
heights_increase

#> # A tibble: 22 x 9
#>   country    max increase  year n_obs continent height_cm year0 country_fct
#>   <chr>      <dbl> <lgl>  <dbl> <int> <chr>      <dbl> <dbl> <fct>
#> 1 Honduras  168. TRUE    1950     6 Americas    164.    240 Honduras
#> 2 Honduras  168. TRUE    1960     6 Americas    164.    250 Honduras
#> 3 Honduras  168. TRUE    1970     6 Americas    165.    260 Honduras
#> 4 Honduras  168. TRUE    1980     6 Americas    165.    270 Honduras
#> 5 Honduras  168. TRUE    1990     6 Americas    165.    280 Honduras
#> 6 Honduras  168. TRUE    2000     6 Americas    168.    290 Honduras
#> 7 Moldova  174. TRUE    1840     5 Europe     165.    130 Moldova
#> 8 Moldova  174. TRUE    1950     5 Europe     172.    240 Moldova
#> 9 Moldova  174. TRUE    1960     5 Europe     173.    250 Moldova
#> 10 Moldova  174. TRUE    1970     5 Europe     174.    260 Moldova
#> # ... with 12 more rows
```

Or tallest country

```
heights_top <- heights_max_in_full %>% top_n(n = 1, wt = max)
heights_top

#> # A tibble: 16 x 9
#>   country    max increase  year n_obs continent height_cm year0 country_fct
#>   <chr>      <dbl> <lgl>  <dbl> <int> <chr>      <dbl> <dbl> <fct>
#> 1 Denmark  183. FALSE    1820    16 Europe     167.    110 Denmark
#> 2 Denmark  183. FALSE    1830    16 Europe     165.    120 Denmark
#> 3 Denmark  183. FALSE    1850    16 Europe     167.    140 Denmark
#> 4 Denmark  183. FALSE    1860    16 Europe     168.    150 Denmark
#> 5 Denmark  183. FALSE    1870    16 Europe     168.    160 Denmark
#> 6 Denmark  183. FALSE    1880    16 Europe     170.    170 Denmark
#> 7 Denmark  183. FALSE    1890    16 Europe     169.    180 Denmark
#> 8 Denmark  183. FALSE    1900    16 Europe     170.    190 Denmark
#> 9 Denmark  183. FALSE    1910    16 Europe     170.    200 Denmark
#> 10 Denmark  183. FALSE    1920    16 Europe     174.    210 Denmark
#> 11 Denmark  183. FALSE    1930    16 Europe     174.    220 Denmark
#> 12 Denmark  183. FALSE    1940    16 Europe     176.    230 Denmark
#> 13 Denmark  183. FALSE    1950    16 Europe     180.    240 Denmark
#> 14 Denmark  183. FALSE    1960    16 Europe     180.    250 Denmark
#> 15 Denmark  183. FALSE    1970    16 Europe     181.    260 Denmark
#> 16 Denmark  183. FALSE    1980    16 Europe     183.    270 Denmark
```

We can then display the data by highlighting it in the background, first creating a background plot and overlaying the plots on top of this as an additional ggplot layer, in Figure 10.

7 Dancing with Models

These same workflows can be used to interpret and explore a model. As the data tends to follow a non linear trajectory, we use a general additive model (gam) with the **mgcv** R package (Wood 2017) using the code below:

```
heights_gam <- gam(
  height_cm ~ s(year0, by = country_fct) + country_fct,
  data = heights_brolgar,
  method = "REML"
)
```

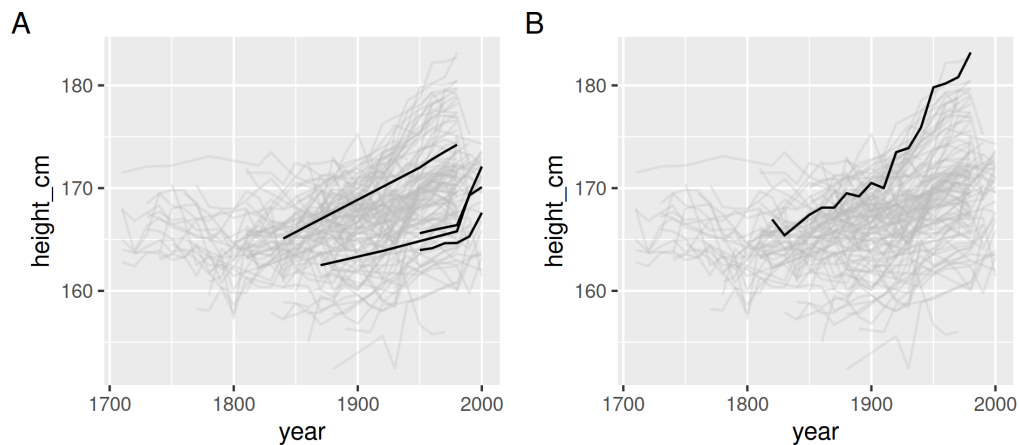


Figure 10: Plots of the data in the background, with the countries that always increase in height through time in A, and the country with the tallest people in B

This fits height in centimetres with a smooth effect for year for each country, with a different intercept for each country. It is roughly equivalent to a random intercept varying slope model. Note that this gam model took approximately 8074 seconds to fit. We add the predicted and residual values for the model below, as well as the residual sums of squares for each country.

```
library(mgcv)
library(modelr)
heights_aug <- heights_brolgar %>%
  add_predictions(heights_gam, var = "pred") %>%
  add_residuals(heights_gam, var = "res") %>%
  group_by_key() %>%
  mutate(rss = sum(res^2)) %>%
  ungroup()
```

We can use the previous approach to explore the model results. We can take a look at a sample of the predictions along with the data, by using `sample_n_keys`. This provides a useful way to explore some set of the model predictions. In order to find those predictions that best summarise the best, and worst, and in between, we need to use the methods in the next section, “Stereotyping”.

```
heights_aug %>%
  sample_n_keys(12) %>%
  ggplot(aes(x = year,
             y = pred,
             group = country)) +
  geom_line(colour = "steelblue") +
  geom_point(aes(y = height_cm)) +
  facet_wrap(~country)
```

8 Stereotyping

To help understand a population of measurements over time, it can be useful to understand which individual measurements are typical (or “stereotypical”) of a measurement. For example, to understand which individuals are stereotypical of a statistic such as the minimum, median, and maximum height. This section discusses how to find these stereotypes in the data.

Figure 12 shows the residuals of the simple model fit to the data in the previous section. There is an overlaid five number summary, showing the minimum, 1st quantile, median, 3rd quantile, and maximum residual value residuals, as well as a rug plot to show the data. We can use these residuals to understand the stereotypes of the data - those individuals in the model that best match to this five number summary.

We can do this using `keys_near()` from `brolgar`. By default this uses the 5 number summary, but any function can be used. You specify the variable you want to find the keys nearest, in this case `rss`, residual sums of squares for each key:

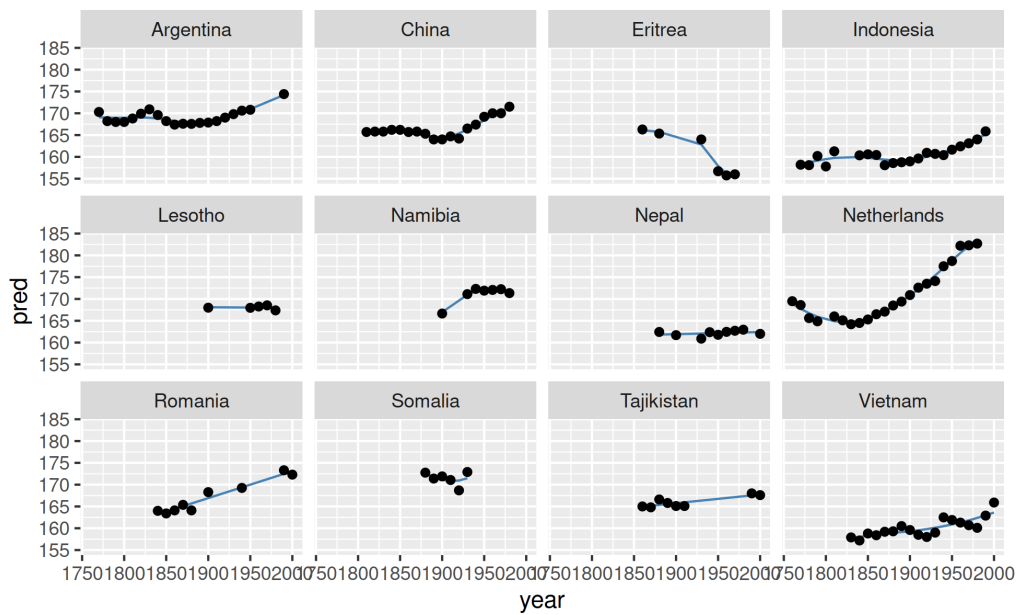


Figure 11: Exploration of a random sample of the data. This shows the data points of 12 countries, with the model fit in blue.

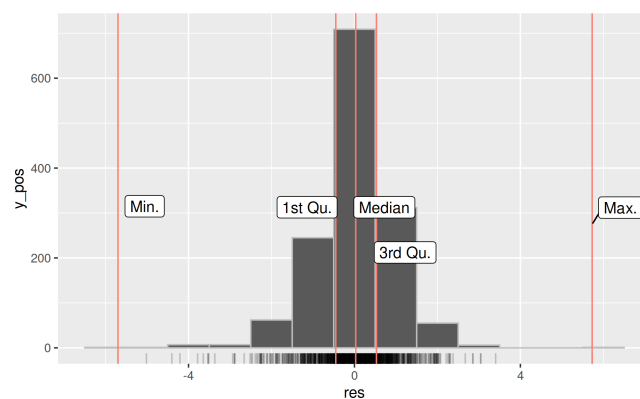


Figure 12: Five number summary of residual values from the model fit. The residuals are centered around zero with some variation.

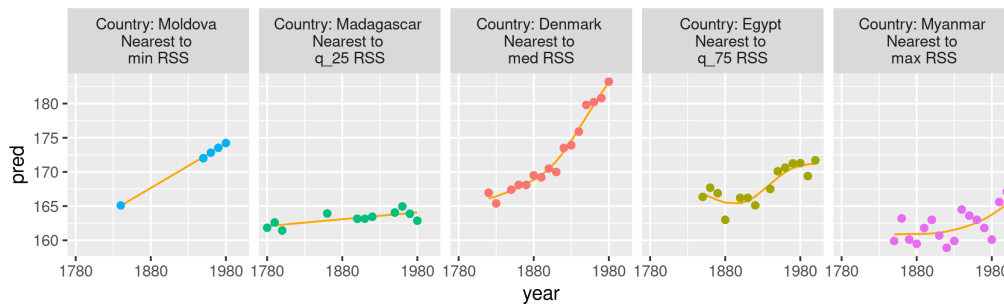


Figure 13: The keys nearest to the five number summary of the residual sums of squares. Moldova and Madagascar are well fit by the model, and are fit by a straight line. The remaining countries with poorer fit have greater variation in height. It is not clear how a better model fit could be achieved.

```
keys_near(heights_aug, var = rss)
```

```
#> # A tibble: 62 x 5
#>   country  rss stat  stat_value stat_diff
#>   <chr>   <dbl> <fct>    <dbl>     <dbl>
#> 1 Denmark  9.54 med      9.54         0
#> 2 Denmark  9.54 med      9.54         0
#> 3 Denmark  9.54 med      9.54         0
#> 4 Denmark  9.54 med      9.54         0
#> 5 Denmark  9.54 med      9.54         0
#> 6 Denmark  9.54 med      9.54         0
#> 7 Denmark  9.54 med      9.54         0
#> 8 Denmark  9.54 med      9.54         0
#> 9 Denmark  9.54 med      9.54         0
#> 10 Denmark 9.54 med      9.54         0
#> # ... with 52 more rows
```

To plot the data, they need to be joined back to the original data, we use a left join, joining by country.

```
heights_near_aug <- heights_aug %>%
  keys_near(var = rss) %>%
  left_join(heights_aug,
    by = c("country"))
```

Figure 13 shows those countries closest to the five number summary. Observing this, we see that the minimum RSS for Moldova fits a nearly perfectly straight line, and the maximum residuals for Myanmar have wide spread of values.

```
ggplot(heights_near_aug,
  aes(x = year,
    y = pred,
    group = country,
    colour = country)) +
  geom_line(colour = "orange") +
  geom_point(aes(y = height_cm)) +
  scale_x_continuous(breaks = c(1780, 1880, 1980)) +
  facet_wrap(~stat + country,
    labeller = label_glue("Country: {country} \nNearest to \n{stat} RSS"),
    nrow = 1) +
  theme(legend.position = "none",
    aspect.ratio = 1)
```

We can also look at the highest and lowest 3 residual sums of squares:

```
heights_near_aug_top_3 <- heights_aug %>%
  distinct(country, rss) %>%
  top_n(n = 3,
```

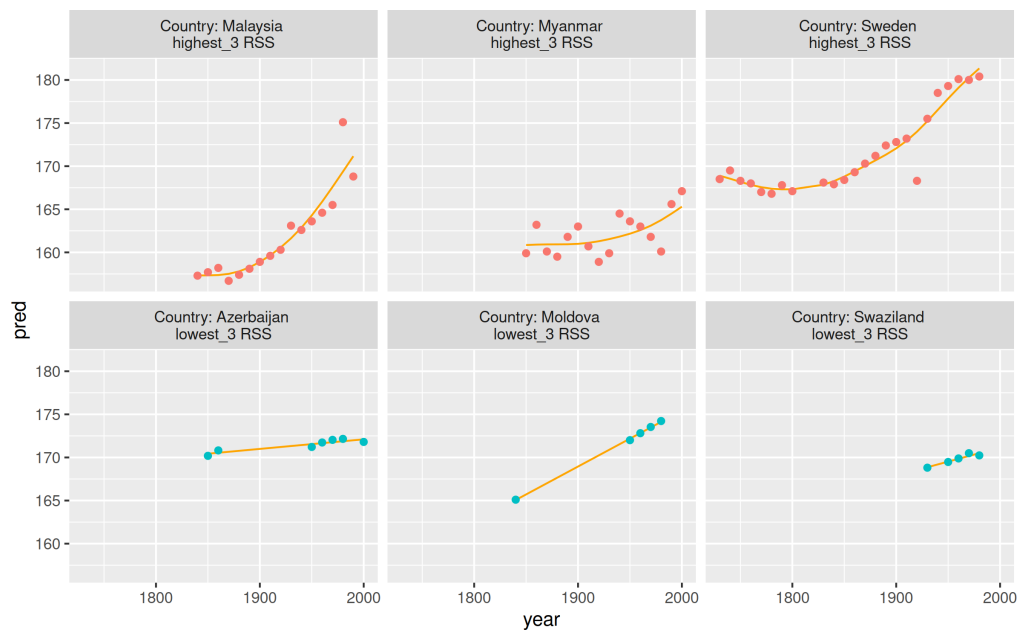


Figure 14: Figure of stereotypes for those keys with the three highest and lowest RSS values. Those that fit best tend to be linear, but those that fit worst have wider variation in heights.

```

    wt = rss)

heights_near_aug_bottom_3 <- heights_aug %>%
  distinct(country, rss) %>%
  top_n(n = -3,
        wt = rss)

heights_near_top_bot_3 <- bind_rows(highest_3 = heights_near_aug_top_3,
                                   lowest_3 = heights_near_aug_bottom_3,
                                   .id = "rank") %>%
  left_join(heights_aug,
            by = c("country", "rss"))

```

Figure 14 shows the same information as the previous plot, but with the 3 representative countries for each statistic. This gives us more data on what the stereotypically “good” and “poor” fitting countries to this model.

9 Getting Started

The **brlgar** R package can be installed from CRAN using

```

# From CRAN
install.packages("brlgar")
# Development version
remotes::install_github("njtierney/brlgar")

```

The functions are all designed to build upon existing packages, but are predicated on working with **tsibble**. The package extends upon **ggplot2** to provide facets for exploration: `facet_sample()` and `facet_strata()`. Extending **dplyr**'s `sample_n()` and `sample_frac()` functions by providing sampling and stratifying based around keys: `sample_n_keys()`, `sample_frac_keys()`, and `stratify_keys()`. New functions are focussed around the use of key, for example `key_slope()` to find the slope of each key, and `keys_near()` to find those keys near a summary statistic. Finally, feature calculation is provided by building upon the existing time series feature package, **feasts**.

To get started with **brlgar** you must first ensure your data is specified as a **tsibble** - discussed earlier in the paper, there is also a vignette “[Longitudinal Data Structures](#)”, which discusses these ideas. The next step we recommend is sampling some of your data with `facet_sample()`, and

`facet_strata()`. When using `facet_strata()`, facets can be arranged in order of a variable, using the `along` argument, which can reveal interesting features.

To further explore longitudinal data, we recommend finding summary features of each variable with `features`, and identifying variables that are near summary statistics, using `keys_near` to find individuals stereotypical of a statistical value.

10 Concluding Remarks

The **brolgar** package facilitates exploring longitudinal data in R. It builds upon existing infrastructure from **tsibble**, and **feasts**, which work within the tidyverse family of R packages, as well as the newer, **tidyverts**, time series packages. Users familiar with either of these package families will find a lot of similarity in their use, and first time users will be able to easily transition from **brolgar** to the tidyverse or **tidyverts**.

Visualizing categorical or binary data over a time period can be difficult as the limited number of values on the y axis leads to overplotting. This can conceal the number of values present at a given value. The tools discussed in **brolgar** facilitate this in the form of `facet_sample`, and `facet_strata`. Some special methods could be developed to add jitter or noise around these values on the y axis, while still maintaining the graphical axis and tick marks.

Future work will explore more features and stratifications, and stereotypes, and generalise the tools to work for data without time components, and other data types.

11 Acknowledgements

We would like to thank Stuart Lee, Mitchell O'Hara Wild, Earo Wang, and Miles McBain for their discussion on the design of **brolgar**. We would also like to thank Rob Hyndman, Monash University and ACEMS for their support of this research.

12 Paper Source

The complete source files for the paper can be found at <https://github.com/njtierney/rjournal-brolgar>. The paper is built using **rmarkdown**, **targets** and **capsule** to ensure R package versions are the same. See the README file on the github repository for details on recreating the paper.

References

- Huntington-Klein, Nick, and Philip Khor. 2020. *Pmdplyr: 'Dplyr' Extension for Common Panel Data Maneuvers*. <https://CRAN.R-project.org/package=pmdplyr>.
- Hyndman, Robin John, and George Athanasopoulos. 2018. *Forecasting: Principles and Practice*. 2nd ed. Australia: OTexts.
- Long, Jacob A. 2020. *Panelr: Regression Models and Utilities for Repeated Measures and Panel Data*. <https://cran.r-project.org/package=panelr>.
- Millo, Giovanni. 2017. "Robust Standard Error Estimators for Panel Models: A Unifying Approach." *Journal of Statistical Software* 82 (3): 1–27. <https://doi.org/10.18637/jss.v082.i03>.
- O'Hara-Wild, Mitchell, Rob Hyndman, and Earo Wang. 2020a. *Fable: Forecasting Models for Tidy Time Series*. <https://CRAN.R-project.org/package=fable>.
- . 2020b. *Feasts: Feature Extraction and Statistics for Time Series*. <https://CRAN.R-project.org/package=feasts>.
- Ryan, Jeffrey A., and Joshua M. Ulrich. 2020. *Xts: eXtensible Time Series*. <https://CRAN.R-project.org/package=xts>.
- Wang, Earo, Dianne Cook, and Rob J. Hyndman. 2020. "A New Tidy Data Structure to Support Exploration and Modeling of Temporal Data." *Journal of Computational and Graphical Statistics* 29 (3): 466–78. <https://doi.org/10.1080/10618600.2019.1695624>.
- Wood, S. N. 2017. *Generalized Additive Models: An Introduction with r*. 2nd ed. Chapman; Hall/CRC.
- Zeileis, Achim, and Gabor Grothendieck. 2005. "Zoo: S3 Infrastructure for Regular and Irregular Time Series." *Journal of Statistical Software* 14 (6): 1–27. <https://doi.org/10.18637/jss.v014.i06>.

Nicholas Tierney
Monash University

Department of Econometrics and Business Statistics
Telethon Kids Institute
Perth, Western Australia
<https://njtierney.com>
ORCID: 0000-0003-1460-8722
nicholas.tierney@gmail.com

Dianne Cook
Monash University
Department of Econometrics and Business Statistics
<https://dicook.org>
ORCID: 0000-0002-3813-7155
dicook@monash.edu

Tania Prvan
Macquarie University
Department of Mathematics and Statistics
ORCID: 0000-0002-6403-4344
tania.prvan@mq.edu.au

The Concordance Test, an Alternative to Kruskal-Wallis Based on the Kendall- τ Distance: An R Package

by Javier Alcaraz, Laura Anton-Sanchez and Juan Francisco Monge

Abstract The Kendall rank correlation coefficient, based on the Kendall- τ distance, is used to measure the ordinal association between two measurements. In this paper, we introduce a new coefficient also based on the Kendall- τ distance, the Concordance coefficient, and a test to measure whether different samples come from the same distribution. This work also presents a new R package, **ConcordanceTest**, with the implementation of the proposed coefficient. We illustrate the use of the Concordance coefficient to measure the ordinal association between quantity and quality measures when two or more samples are considered. In this sense, the Concordance coefficient can be seen as a generalization of the Kendall rank correlation coefficient and an alternative to the non-parametric mean rank-based methods for comparing two or more samples. A comparison of the proposed Concordance coefficient and the classical Kruskal-Wallis statistic is presented through a comparison of the exact distributions of both statistics.

1 Introduction

When we have a sample of observations of a given population it may be difficult to assume that they come from a certain distribution since we may not always have any type of information about the variable under study and when we do, it may not be enough to determine the type of distribution. In these cases, parametric inference is inappropriate. Moreover, this type of technique may be unsuitable should the observations not fulfill any of the basic assumptions on which they are based; normality and a large quantity of data.

Violation of the necessary assumptions in parametric statistics necessitates the use of non-parametric statistics. Non-parametric tests do not depend on the definition of a distribution function or statistical parameters such as mean, variance, etc. The use of non-parametric tests, despite being less powerful, is also adequate when there are not enough observations available, when data are non-normal data or when ordinal data are being analyzed.

Although the first steps in non-parametric statistics began earlier, it was not until the 1930s that a systematic study in this field appeared. Fisher (1935) introduced the permutation test or randomization test as a simple way to compute the sampling distribution for any test statistic under the null hypothesis that does not establish any effect on all possible outcomes. Over the next two decades some of the main non-parametric tests emerged, Friedman (1940); Kendall and Smith (1939); Kendall (1938); Kruskal (1958); Kruskal and Wallis (1952); Mann and Whitney (1947); Pitman (1937); Wilcoxon (1945), among others.

The main advantages of the non-parametric tests are: the data can be nonnumerical observations while they can be classified according to some criterion, they are usually easy to calculate and do not make any hypothesis about the distribution of the population from which the samples are taken. We can also cite two drawbacks: the non-parametric tests are less precise than other statistical models and they are based on the order of the elements in the sample and this order will likely stay the same even if the numerical data change.

There are many non-parametric tests in the literature, which can basically be classified into four categories depending on whether: it is a test to compare two or more than two related samples or a test for comparing related or unrelated samples. Examples of the most used non-parametric tests in the literature for each of these four situations are the following: the *Wilcoxon signed-rank test* (Wilcoxon, 1945) for comparing two related samples, the *Mann-Whitney (Wilcoxon) test* (Mann and Whitney, 1947) for comparing two unrelated samples, the *Friedman test* (Friedman, 1940) for comparing three or more related samples, and the *Kruskal-Wallis test* (Kruskal and Wallis, 1952) for comparing three or more unrelated samples. Several methods that exploit some characteristic of the samples have appeared in the literature in recent years, such as Alhakim and Hooper (2008); Terpstra and Magel (2003).

It is also possible to measure the degree of association of two variables through a non-parametric approach, in that sense we can mention the Kendall rank correlation coefficient (Kendall, 1938) and the Spearman rank correlation coefficient (Spearman, 1904).

In Aparicio et al. (2020), the authors introduce the *Kendall- τ partition ranking*; given a ranking of elements of a set and given a disjoint partition of the same set, the Kendall- τ partition ranking is the

induced linear order of the subsets of the partition which follows from the given ranking of elements of a set. In this work, we propose to use the Kendall- τ distance as a concordance measure between the different samples in an ordered set of observations. In this regard, the proposed measure, which we call *Concordance coefficient*, can be considered as an extension of the Kendall rank correlation coefficient when more than two samples are considered. The main difference between the proposed measure and the previous ones, is the consideration of the Kendall- τ distance instead of ranks, which use classical methods. We also propose a significance test in order to determine when more than two samples come from the same distribution, and present a comparison with the classical Kruskal-Wallis method. We illustrate the use of the proposed coefficient with a new R package, **ConcordanceTest** (Alcaraz et al., 2022), which is freely available from the Comprehensive R Archive Network (CRAN). Actually, R establishes the state of the art in statistical software. There are currently packages for all the non-parametric tests mentioned above, for example: the **Kendall** package (McLeod, 2011), which deals with the Kendall rank correlation coefficient; the **pspearman** package (Savicky, 2014), with the Spearman rank correlation coefficient; or the **stats** package: an R Core Team and contributors' worldwide package that contains many of the non-parametric tests for comparing two or more, related or unrelated, samples. The Kendall- τ distance, on which the proposed coefficient is based, is one of the most used in distance-based models, for which there are also recent alternatives in R. See, for example, the **PerMallows** (Irurozki et al., 2016), **rankdist** (Qian and Yu, 2019) or **BayesMallows** packages (Sorensen et al., 2020).

The remainder of this paper is organized as follows. After a brief review in the next section of the main features of the Kendall rank correlation coefficient and the Kruskal-Wallis statistic, in the following two sections we present the coefficient we propose in this work and illustrate its use with our **ConcordanceTest** package. Specifically, in the third section we introduce the Concordance coefficient while in the fourth section the related statistical test is presented. The fifth section includes a comparison between the Kruskal-Wallis test in the **stats** package and that presented in this work. Some final remarks follow in the last section. Appendix A presents an example of the probability distribution of the Concordance coefficient and the Kruskal-Wallis statistic. Appendix B deals with a comparison between the probability density function of the Concordance coefficient and the Kruskal-Wallis statistic for several experiments. Finally, Appendix C presents some details of how the p-values for the Concordance coefficient have been calculated and shows some critical values and exact p-values.

2 Non-parametric tests

This section presents the Kendall rank correlation coefficient (Kendall, 1938), a coefficient to measure the relationship between two samples ordinally, and the Kruskal-Wallis statistical test (Kruskal and Wallis, 1952), which is a rank-based statistical test to measure whether different samples come from the same distribution, without assuming a given distribution for the population.

Only these two non-parametric tests are presented in detail, since the test proposed in this paper uses the Kendall- τ distance and it can be seen as an extension of the Kendall rank correlation coefficient when more than two samples are considered, and it is presented as an alternative to the Kruskal-Wallis statistical test.

The Kendall rank correlation coefficient is a non-parametric measure of correlation. This measure is based on the Kendall- τ distance between two permutations of n elements. The Kendall- τ distance ($d_{K-\tau}$) is defined as the number of pairwise disagreements between two permutations π_1 and π_2 . For instance, if we have three elements, the distance from permutation 123 to permutations 132, 231 and 321 is 1, 2 and 3 respectively. The maximum number of disagreements that may occur between two permutations of n elements is $n(n-1)/2$ and, in this case, all the values of permutation π_1 are in the reverse order of π_2 .

The Kendall rank correlation coefficient between permutations π_1 and π_2 , denoted by τ , is defined by

$$\tau = 1 - 2 \frac{d_{K-\tau}(\pi_1, \pi_2)}{n(n-1)/2}.$$

The Kendall rank correlation coefficient is used as a statistical test to determine whether there is a relationship or dependence between two random variables. The main advantages of this coefficient are: the data can be non-numerical observations if they can be ordered, it is easy to calculate, and the associated statistical test does not assume a known distribution of the population from which the samples are taken.

The Kruskal-Wallis test is a non-parametric statistical method to study whether different samples come from the same population. The test is the extension of the Mann-Whitney Test (Mann and Whitney, 1947) when we have more than two samples or groups. The following example illustrates the Kruskal-Wallis test when comparing three samples.

Example 1 Let us assume that the effectiveness of three different treatments (A, B, C) has been measured for 6 individuals, two individuals being assigned to each of the treatments, with the effectiveness of each treatment being measured ordinally. We could obtain the result shown in Table 1, where, for example, the effectiveness of treatment A has been rated in first and third place.

	A	B	A	C	C	B
Rank	1	2	3	4	5	6

Table 1: Result for an experiment with 6 people and 3 treatments.

The Kruskal-Wallis statistic is determined by the difference between the ranks of the individuals in each category with the average rank. In our example, the average rank of the test is $\bar{R} = 3.5$, while the average rank of each of the three treatments are $\bar{R}_A = 2$, $\bar{R}_B = 4$ and $\bar{R}_C = 4.5$. The Kruskal-Wallis statistic, denoted by H , is based on the calculation of the distance of each rank to the average rank, which can be expressed as follows:

$$H = -3(n + 1) + \frac{12}{n(n + 1)} \sum_{i=1}^k \frac{R_i^2}{n_i},$$

where n is the number of observations in the k samples, n_i is the number of observations in the i -th sample and R_i is the sum of the ranks in the i -th sample. In our example, the value of the Kruskal-Wallis statistic is:

$$H = -3(n + 1) + \frac{12}{n(n + 1)} \sum_{i=1}^k \frac{R_i^2}{n_i} = -3(6 + 1) + \frac{12}{6(6 + 1)} \left(\frac{4^2}{2} + \frac{8^2}{2} + \frac{9^2}{2} \right) = 2.$$

Table 2 shows the probability distribution of the Kruskal-Wallis statistic for 3 treatments, each with 2 patients. Appendix A presents the Kruskal-Wallis statistic for all possible results in the experiment for 3 treatments with 2 people in each. In [Spurrer \(2003\)](#), the author compares different methods for approximating the null probability points.

H	$Prob$
0.00	0.06667
0.29	0.13333
0.86	0.13333
1.14	0.13333
2.00	0.13333
2.57	0.06667
3.43	0.13333
3.71	0.13333
4.57	0.06667

Table 2: Probability distribution for the Kruskal-Wallis statistic (H), with sample sizes $N = (2, 2, 2)$.

3 The Concordance coefficient τ_c

In [Aparicio et al. \(2020\)](#), the authors introduce the *Kendall- τ partition ranking*; given a ranking of elements of a set and given a disjoint partition of the same set, the Kendall- τ partition ranking is the induced linear order of the subsets of the partition which follows from the given ranking of elements of a set. The Kendall- τ partition ranking presents an ordinal alternative to the mean-based ranking that uses a pseudo-cardinal scale. Let π be permutation of the elements of set V and let V_1, V_2, \dots, V_k be a partition of V then, the Kendall- τ distance from permutation π is given by

$$d_{K-\tau} = \min\{d_{K-\tau}(\rho, \pi) : \text{elements in } V_r \text{ are consecutively listed in } \rho, \forall r\}.$$

This distance is also called the disorder of permutation π . For the calculation of the disorder of a permutation of elements, in [Aparicio et al. \(2020\)](#), the authors establish that the distance or disorder of a permutation of elements $\pi = (a|a|b|b|a|c|a|b|c| \dots |c|a|b)$ is given by the solution of the Linear

Ordering Problem (LOP) with the preference matrix M , where the element m_{ab} of matrix M indicates the number of times that an element a of sample A precedes an element b of sample B in the order π . The solution of the Linear Ordering Problem gives us a new order in the elements of π , the closest to π , in which all the elements belonging to the same sample are listed consecutively. The book publication by [Martí and Reinelt \(2011\)](#) provides an exhaustive study of the Linear Ordering Problem.

The authors [Aparicio et al. \(2020\)](#) present the properties of the Kendall- τ partition ranking and compare it with classical mean and median-based rank approaches. Those properties are extracted from social choice theory and are adapted to a partition ranking, see [Arrow \(1951\)](#); [Kemeny \(1959\)](#); [Zahid and Swart \(2015\)](#). Two of these properties are only true for the Kendall- τ partition ranking: the *Condorcet* and *Deletion Independence* properties. The Condorcet property establishes that the most preferred subset must be listed before any other in any ranking; and the Deletion Independence property establishes that if any subset is removed, then the induced order of subsets does not change. In permutation $\pi = (c|c|c|b|b|a|a|c|c)$ the set C is a condorcet winner, the most preferred set, but B has a lesser mean rank value than set C if set A is not considered in the comparison; therefore, the permutation $\pi = (c|c|c|b|b|a|a|c|c)$ gives an example where ranking subsets from ranks is not very reliable.

From [Aparicio et al. \(2020\)](#), the maximum number of disagreements that may occur in a permutation of n elements (where the elements are classified in k subsets V_1, V_2, \dots, V_k of sizes n_1, n_2, \dots, n_k respectively) is $\sum_{r=1}^k \sum_{s=r+1}^k n_r n_s - (GP_b + \sum_{r=1}^k \sum_{s=r+1}^k \lfloor \frac{n_r n_s}{2} \rfloor)$, where GP_b is the Generalized Pentagonal Number of b , and b the number of subsets V_k with odd cardinality. The Generalized Pentagonal number GP_b , for $b \in \mathbb{N}$, is

$$GP_b = \begin{cases} \frac{\ell(3\ell - 1)}{2} & b = 2\ell \text{ (} b \text{ even),} \\ \frac{\ell(3\ell + 1)}{2} & b = 2\ell + 1 \text{ (} b \text{ odd).} \end{cases}$$

This maximum number of disagreements (the maximum disorder) in a permutation π of elements, allows us to define a relative disorder coefficient of permutation π as

$$relative\ disorder(\pi) = \frac{d_{K-\tau}(\pi)}{\sum_{r=1}^k \sum_{s=r+1}^k n_r n_s - (GP_b + \sum_{r=1}^k \sum_{s=r+1}^k \lfloor \frac{n_r n_s}{2} \rfloor)}$$

Definition 1 We define the Concordance coefficient (τ_c) of permutation π as

$$\tau_c = 1 - relative\ disorder(\pi) = 1 - \frac{d_{K-\tau}(\pi)}{\sum_{r=1}^k \sum_{s=r+1}^k n_r n_s - (GP_b + \sum_{r=1}^k \sum_{s=r+1}^k \lfloor \frac{n_r n_s}{2} \rfloor)}$$

The Concordance coefficient (τ_c) provides a measure of independence in the k samples, where τ_c is a value between 0 and 1, taking the value of 1 when there is a total order between the samples, and 0 when the disorder is maximum. In this sense, the Concordance coefficient can be seen as a generalization of the Kendall rank correlation coefficient when we have more than two samples. Given that the Concordance coefficient satisfies the properties mentioned above, we consider it is more appropriate for measuring differences between samples than a rank-based method, such as Kruskal-Wallis’.

Example 1 (Cont.) Continuing with the data in Example 1, the results of the experiment provide the following order or permutation of the treatments $\pi = (a|b|a|c|c|b|)$.

Given the order of individuals $\pi = (a|b|a|c|c|b|)$, the ordering between individuals that leaves individuals with the same treatment together is ordination $(a\ a\ b\ b\ c\ c)$ or $(a\ a\ c\ c\ b\ b)$. Both ordinations only need 3 pairwise disagreements from the permutation π . In order to find the permutation of elements (equal elements listed consecutively) closer to a given permutation, it is sufficient to solve the Linear Ordering Problem (LOP) with the preference matrix defined above. In this example, said matrix is:

$$\begin{matrix} & A & B & C \\ A & - & 3 & 4 \\ B & 1 & - & 2 \\ C & 0 & 2 & - \end{matrix},$$

where each element of the matrix m_{ij} represents the number of times an individual of a treatment i precedes an individual of the treatment j . The solution of the LOP is the permutation of treatments which maximizes the preferences of order in the experiment, that is, in this example, the permutations of treatments $(A B C)$ or $(A C B)$ retain 9 preferences expressed in the order of individuals represented by the permutation π . Therefore, the distance of the permutation π to a total order between treatments is $\sum_{i < j} n_i n_j - 9 = 3$. This distance, which is the number of pairwise disagreements needed in a permutation of elements to reach a permutation that establishes a total order between treatments, is denominated the disorder of a permutation by the authors of the work by Aparicio et al. (2020)¹.

Then, the relative disorder of permutation π can be evaluated as

$$relative\ disorder(\pi) = \frac{d_{K-\tau}(\pi)}{\sum_{r=1}^k \sum_{s=r+1}^k n_r n_s - (GP_b + \sum_{r=1}^k \sum_{s=r+1}^k \lfloor \frac{n_r n_s}{2} \rfloor)} = \frac{3}{12 - (0 + 6)} = \frac{3}{6} = \frac{1}{2}$$

and the Concordance coefficient

$$\tau_c = 1 - relative\ disorder(\pi) = 1 - \frac{1}{2} = \frac{1}{2}$$

Notice that no set of this example has odd cardinality, therefore the pentagonal number is $GP_0 = 0$.

Table 3 shows the probability distribution of the disorder and the Concordance coefficient for 3 treatments with 2 patients each. Appendix A presents the disorder and the Concordance coefficient for all possible results in the experiment with sample sizes $N = (2, 2, 2)$. Figure 1 compares the probability distribution of the Concordance coefficient and the Kruskal-Wallis statistic, for 3 treatments and 2 people in each treatment. Notice that some Kruskal-Wallis statistic values ($H=2.57$) are less probable than large ones.

<i>dis</i>	τ_c	<i>Prob</i>
6	0.0000	0.06667
5	0.1667	0.13333
4	0.3333	0.20000
3	0.5000	0.20000
2	0.6667	0.20000
1	0.8333	0.13333
0	1.0000	0.06667

Table 3: Probability distribution of the disorder (*dis*) and the Concordance coefficient (τ_c), with sample sizes $N = (2, 2, 2)$.

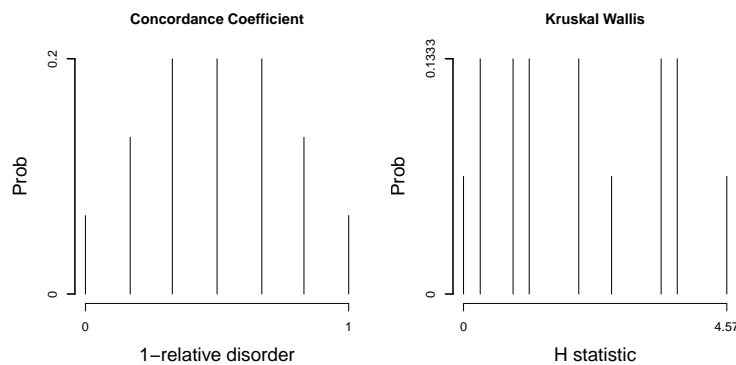


Figure 1: Probability distribution of the Concordance coefficient ($\tau_c=1$ -relative disorder) and the Kruskal-Wallis statistic (H), with sample sizes $N = (2, 2, 2)$.

¹If the number of samples is small, we can evaluate all the possibilities in order to obtain the solution of the Linear Ordering Problem, for example, if we have 3 samples the number of feasible solutions for the LOP is $3! = 6$.

The Concordance coefficient in `ConcordanceTest` package

The R package we have developed allows to calculate both the Concordance coefficient and the Kruskal-Wallis statistic in order to facilitate their comparison. Given the high combinatorial degree of the problem of ordering samples of populations, some of the functions implemented in the package can perform the calculations exactly, exploring the entire sample space or possibilities, or they can approximate the sample space or possibilities by simulation.

The `ConcordanceTest` package can be installed from CRAN:

```
install.packages("ConcordanceTest")
library("ConcordanceTest")
```

and its functions can perform the calculations related only to the Concordance coefficient (default option, specified with the parameter $H=0$) or do them also for the Kruskal-Wallis statistic ($H=1$), allowing their comparison.

To obtain the probability distribution of the statistics, it is necessary to have the set of all possible permutations that can occur in the result of the experiment that we want to analyze ($90=6!/2!2!2!$ in Example 1). This can be obtained through the function `Permutations_With_Repetition()`, which has been developed and included in the `ConcordanceTest` package.

The function `CT_Distribution()` calculates the probability distribution of the Concordance coefficient and the Kruskal-Wallis statistic. The set of possibilities (sample space) grows very quickly with the number of elements and with the number of sets and, in some cases, to calculate the probability distribution in an exact way becomes unaffordable, making it necessary to approximate calculations. Both an exact and an approximate calculation (default option) can be done using the function `CT_Distribution()`. It is used as follows:

```
CT_Distribution(Sample_Sizes, Num_Sim = 10000, H = 0, verbose = TRUE)
```

where `Sample_Sizes` is a numeric vector (n_1, \dots, n_k) containing the number of repetitions of each element, i.e., the size of each sample in the experiment. `Num_Sim` is the number of simulations to be performed in order to obtain the probability distribution of the statistics (10,000 by default). If `Num_Sim` is set to 0, the probability distribution tables are obtained exactly using the function `Permutations_With_Repetition()`. `H` is the parameter specifying whether the calculations must also be performed for the Kruskal-Wallis statistic, and `verbose` is a logical parameter that indicates whether some progress report of the simulations should be given.

Example 1 (Cont.) *Using the function `CT_Distribution()` with `Num_Sim` equal to 0, we could obtain the probability distribution of the Kruskal-Wallis statistic and the Concordance coefficient in Example 1 (Tables 2 and 3, respectively) in an exact way. As shown in this example, we can also approximate the probability distributions of Example 1 by simulating, for example, 25,000 permutations of 3 treatments with 2 patients each. Note that, for reproducibility, we always initialize the generator for pseudo-random numbers when the results rely on simulation.*

```
set.seed(12)
Sample_Sizes <- c(2,2,2)
CT_Distribution(Sample_Sizes, Num_Sim = 25000, H = 1)
```

```
$C_freq
  disorder Concordance coefficient Frequency Probability
[1,]      6              0.00           6      0.0667
[2,]      5              0.17          12      0.1333
[3,]      4              0.33          18      0.2000
[4,]      3              0.50          18      0.2000
[5,]      2              0.67          18      0.2000
[6,]      1              0.83          12      0.1333
[7,]      0              1.00           6      0.0667
```

```
$H_freq
  H Statistic Frequency Probability
[1,]      0.00           6      0.0667
[2,]      0.29          12      0.1333
[3,]      0.86          12      0.1333
[4,]      1.14          12      0.1333
[5,]      2.00          12      0.1333
[6,]      2.57           6      0.0667
```

[7,]	3.43	12	0.1333
[8,]	3.71	12	0.1333
[9,]	4.57	6	0.0667

The function `CT_Distribution()` returns two elements. `C_freq` is a matrix with the probability distribution of the Concordance coefficient. Each row in the matrix contains the disorder, the value of the Concordance coefficient τ_c , the frequency and its probability. `H_freq` (only returned if $H = 1$) is a matrix with the probability distribution of the Kruskal-Wallis statistic. Each row in the matrix contains the value of the statistic H , the frequency and its probability. The results obtained by the function `CT_Distribution()` are the same as those previously shown in Table 3 and Table 2 of Example 1.

4 Concordance test

In this section, we present the Concordance test in order to evaluate when different samples come from the same population distribution. The randomization test introduced by Fisher (1935) establishes a framework for the statistical test based on permutations, see also Box (1980); Stern (1990); Welch (1990).

If all the samples come from the same distribution, then all possible ways to rank n observations divided into k samples have the same probability of occurring. If a result of the experiment provides an order of the observations with a high disorder, it will support the idea that all observations come from the same population. On the contrary, a result with a small disorder will go against the claim that the observations come from the same population. In this way, we propose to consider samples that come from the same distribution as null hypothesis, while the alternative hypothesis is that some of the samples come from a different distribution.

H_0 : There is no difference among the k populations.

H_a : At least one of the populations differs from the other populations.

The decision rule is to reject the null hypothesis if the disorder in the permutation of observations is small, equivalently if the Concordance coefficient τ_c is close to one. We reject the null hypothesis H_0 at the significance level α if τ_c is greater than the percentile $(1 - \alpha)\%$ of the probability distribution of τ_c .

The following example illustrates the use of the Concordance test proposed in this work and compares it with the classical Kruskal-Wallis non-parametric test. The comparison will be made first considering that there are no ties and then modifying the data in the example so that ties appear.

Example 2 Suppose we have applied three treatments to 18 patients, measuring the number of hours it takes these patients to recover. The results are shown in Table 4.

	Hours									
Treatment A	12	13	15	20	23	28	30	32	40	48
Treatment B	29	31	49	52	54					
Treatment C	24	26	44							

Table 4: Result for an experiment with 18 patients and 3 treatments.

Concordance test:

The experiment ranks the patients in the following ranking

$$(a a a a c c a b a b a a c a b b b).$$

If we perform the contrast using the disorder statistic or the Concordance coefficient τ_c , we must calculate the permutation of treatments that maximizes the order between patients obtained in the experiment. The matrix of preferences between treatments observed is as follows:

$$\begin{matrix} & A & B & C \\ A & - & 43 & 19 \\ B & 7 & - & 2 \\ C & 11 & 13 & - \end{matrix}$$

The order between treatments that maximizes the order between patients corresponds to the order (A C B), satisfying 75 of the 95 preferences contained in the matrix, where the value 75 is the solution of the Linear Ordering Problem (LOP)². Therefore, exactly 20 = 95-75 is the number of pairwise disagreements necessary to order the samples and obtain the order (ACB), that is, the disorder is 20. The greatest disorder that an order of elements can have with samples of 10, 5 and 3 elements is given by: $\sum_{r=1}^3 \sum_{s=r+1}^3 n_r n_s - (GP_b + \sum_{r=1}^3 \sum_{s=r+1}^3 \lfloor \frac{n_r n_s}{2} \rfloor) = 95 - (1 + 47) = 47$, therefore the Concordance coefficient is $\tau_c = 1 - 20/47 = 0.574$. The p-value of the disorder 20 or, equivalently, of the Concordance coefficient $\tau_c = 0.574$ is 0.049272³, therefore, at a level of significance less than 5% we can reject the null hypothesis of equality in treatments.

Kruskal-Wallis test:

The treatments A, B and C have average ranks of 7.3, 14.2 and 9, respectively, and the sum of ranks are $R_A = 73, R_B = 71$ and $R_C = 27$.

The Kruskal-Wallis statistic is given by:

$$H = -3(n + 1) + \frac{12}{n(n + 1)} \sum \frac{R_i^2}{n_i} = -3(18 + 1) + \frac{12}{18(18 + 1)} \left(\frac{73^2}{10} + \frac{71^2}{5} + \frac{27^2}{3} \right) = 5.6$$

In Meyer and Seaman (2015), exact values for the Kruskal-Wallis contrast at different levels of significance are found. We can conclude by looking at the tables that the p-value of the H statistic is greater than 0.05, therefore, we cannot reject the null hypothesis that the treatments are equally effective.

Comparing both methods, the Concordance and Kruskal-Wallis tests provide similar results about the statistic but the conclusion differs.

Example 3 Suppose we have the same experiment as in Example 2 but with ties. The results are shown in Table 5. Ties are in bold.

	Hours									
Treatment A	12	13	15	20	24	29	30	32	40	49
Treatment B	29	31	49	52	54					
Treatment C	24	26	44							

Table 5: Result for an experiment with 18 patients and 3 treatments. Example with ties.

Concordance test with ties:

The results of the experiment order the individuals according to the sequence:

$$(a a a a (a c) c (a b) a b a a c (a b) b b)$$

where the elements grouped in the order indicates that they tie. There are 8 different possibilities in order to undo ties in the ranking of elements. If the same probability is assumed for all of them, the expected preference matrix between treatments is given distributing the preference in the comparison of repeated observations with the same weight, that is, assigning the value 0.5 to each of the treatments when two tied units are compared. The preference matrix for this example would be as follows:

$$\begin{matrix} & A & B & C \\ A & \left(\begin{matrix} - & 42 & 18.5 \end{matrix} \right) \\ B & \left(\begin{matrix} 8 & - & 2 \end{matrix} \right) \\ C & \left(\begin{matrix} 11.5 & 13 & - \end{matrix} \right) \end{matrix}$$

Note that the previous matrix represents the matrix of expected preferences if all permutations of items with ties in which they are undone are considered, with the same probability of tie between elements.

²The solution of LOP for this example is the permutation of sets that maximizes the preferences in the preference matrix. It is sufficient to compare the 6 possibilities, (A B C) = 64, (A C B) = 75, (B A C) = 28, (B C A) = 20, (C A B) = 67 and (C B A) = 31.

³Tables of p-values for the Concordance coefficient τ_c are in Appendix C.

The order between treatments that maximizes the order between patients, corresponds to the order $(A C B)$, satisfying 73.5 of the 95 preferences contained in the matrix, where 73.5 is the solution of the Linear Ordering Problem. Therefore, $21.5 = 95 - 73.5$ is the expected number of pairwise disagreements necessary to order the samples and obtain the order $(A C B)$, that is, the disorder is 21.5 or, equivalently, the Concordance coefficient is $\tau = 1 - 21.5/47 = 0.543$, a value with a significance greater than 0.05, $p - value > 0.05$. In this case, the observed data do not show significant evidence in favor of a difference in the effectiveness of treatments.

Kruskal-Wallis test with ties:

The treatments A, B and C have average ranks of 7.45, 14 and 8.83, respectively, and the sum of ranks are $R_A = 74.5$, $R_B = 70$ and $R_C = 26.5$.

The Kruskal-Wallis statistic is given by:

$$H = -3(n+1) + \frac{12}{n(n+1)} \sum \frac{R_i^2}{n_i} = -3(18+1) + \frac{12}{18(18+1)} \left(\frac{74.5^2}{10} + \frac{70^2}{5} + \frac{26.5^2}{3} \right) = 5.074$$

If we make the adjustment in the statistic for ties, we get:

$$\tilde{H} = \frac{H}{1 - \frac{\sum (t_i^3 - t_i)}{N^3 - N}} = \frac{5.074}{1 - \frac{(2^3 - 2) + (2^3 - 2) + (2^3 - 2)}{18^3 - 18}} = 5.0897$$

where t_i is the number of ties of each value.

In this case, the Kruskal-Wallis test provides the same conclusion as the Concordance test; uncertainty being greater when we have ties.

Concordance test in ConcordanceTest package

The **ConcordanceTest** R-package allows to perform the hypothesis test for testing whether samples originate from the same distribution with the function `CT_Hypothesis_Test()`, which carries out the calculations by simulation. It is used as follows:

```
CT_Hypothesis_Test(Sample_List, Num_Sim = 10000, H = 0, verbose = TRUE)
```

where `Sample_List` is a list of numeric data vectors with the elements of each sample, `Num_Sim` is the number of used simulations (10,000 by default), `H` specifies whether the Kruskal-Wallis test must also be done, and `verbose` is a logical parameter that indicates whether some progress report of the simulations should be given.

Example 2 (Cont.) We use the **ConcordanceTest** package to perform the Concordance and Kruskal-Wallis tests of Example 2. We use 25,000 simulations.

```
set.seed(12)
A <- c(12,13,15,20,23,28,30,32,40,48)
B <- c(29,31,49,52,54)
C <- c(24,26,44)
Sample_List <- list(A, B, C)
CT_Hypothesis_Test(Sample_List, Num_Sim = 25000, H = 1)
```

```
$results
              Statistic p-value
Concordance coefficient    0.574 0.04928
Kruskal Wallis             5.600 0.05292
```

```
$C_p_value
[1] 0.04928
```

```
$H_p_value
[1] 0.05292
```

The function `CT_Hypothesis_Test()` provides the value of the statistics together with the p-value associated with each of them. The result of the Kruskal-Wallis test is only returned if `H = 1`. Note that

the approximated p-values obtained by simulation are close to the exact ones, 0.04927 and 0.05223 for the Concordance coefficient and the Kruskal-Wallis statistic, respectively.

An alternative to the contrast performed with the function `CT_Hypothesis_Test()` is to obtain the critical values of our contrast. This can be done with the **ConcordanceTest** package both in an exact or approximate way, using the function `CT_Critical_Values()`. It is used as follows:

```
CT_Critical_Values(Sample_Sizes, Num_Sim = 10000, H = 0, verbose = TRUE)
```

where `Sample_Sizes` is a numeric vector (n_1, \dots, n_k) containing the number of repetitions of each element, i.e., the size of each sample in the experiment. `Num_Sim` is the number of simulations carried out in order to obtain the probability distribution of the statistics (10,000 by default). If `Num_Sim` is set to 0, the critical values are obtained in an exact way. Otherwise they are obtained by simulation. `H` is the parameter specifying whether the critical values of the Kruskal-Wallis test must be calculated and returned, and `verbose` is a logical parameter that indicates whether some progress report of the simulations should be given.

The function returns a list with two elements. `C_results` are the critical values and p-values for a desired significance levels of 0.1, .05 and .01 of the Concordance coefficient, and `H_results` are the critical values and p-values of the Kruskal-Wallis statistic (only returned if `H = 1`).

Example 2 (Cont.) *We show the results of the function `CT_Critical_Values()` with sample sizes $N = (10, 5, 3)$ and 25,000 simulations. The results allow us to compare the test statistics with different significance levels.*

```
set.seed(12)
Sample_Sizes <- c(10, 5, 3)
CT_Critical_Values(Sample_Sizes, Num_Sim = 25000, H = 1)
```

```
$C_results
  | disorder | Concordance coefficient | p-value
Sig level .10      23                0.51    0.0954
Sig level .05      20                0.57    0.0492
Sig level .01      14                0.70    0.0096
```

```
$H_results
  | H Statistic | p-value
Sig level .10      4.55    0.0995
Sig level .05      5.72    0.0497
Sig level .01      7.78    0.0097
```

To obtain the Concordance coefficient and the Kruskal-Wallis statistic from the result of an experiment, the **ConcordanceTest** package has the function `CT_Coefficient()`. This function is useful when we only want to obtain the value of the statistic to check its significance using statistical tables. The function `CT_Coefficient()` is used as follows:

```
CT_Coefficient(Sample_List, H = 0)
```

where `Sample_List` is a list of numeric data vectors with the elements of each sample, and `H` is defined as usual.

Example 2 (Cont.) *We show the results of the function `CT_Coefficient()` for the data in Example 2.*

```
A <- c(12, 13, 15, 20, 23, 28, 30, 32, 40, 48)
B <- c(29, 31, 49, 52, 54)
C <- c(24, 26, 44)
Sample_List <- list(A, B, C)
CT_Coefficient(Sample_List, H = 1)
```

```
$Sample_Sizes
[1] 10 5 3
```

```
$order_elements
[1] 1 1 1 1 1 3 3 1 2 1 2 1 1 3 1 2 2 2
```

```
$disorder
[1] 20
```

```
$Concordance_Coefficient
[1] 0.5744681
```

```
$H_Statistic
[1] 5.6
```

The function `CT_Coefficient()` returns a list with the following elements: `Sample_Sizes` is a numeric vector with the sample sizes, `order_elements` is a numeric vector containing the elements order, `disorder` is the disorder of the permutation given by `order_elements`, `Concordance_Coefficient` is the value of the Concordance coefficient τ_c , that is, 1 minus the relative disorder of the permutation given by `order_elements`, and `H_Statistic` is the Kruskal-Wallis statistic (only returned if $H = 1$).

Note that we can also solve problems with ties (as in Example 3) with the **ConcordanceTest** package.

Other functions in the ConcordanceTest package

The graphical visualization of the probability distributions of the Concordance coefficient and the Kruskal-Wallis statistic can be done with the function `CT_Probability_Plot()`. It is used as follows:

```
CT_Probability_Plot(C_freq = NULL, H_freq = NULL)
```

Using the function `CT_Density_Plot()` of the **ConcordanceTest** package, we can make an approximate representation of the density functions of the statistics, assuming that the probability distributions represent a sample of a continuous variable. It is used as follows:

```
CT_Density_Plot(C_freq = NULL, H_freq = NULL)
```

In both functions, `C_freq` is the probability distribution of the Concordance coefficient and `H_freq` is the probability distribution of the Kruskal-Wallis statistic, obtained exactly or approximately with the function `CT_Distribution()`. The function `CT_Probability_Plot()` can represent both probability distributions or only one of them (if it only receives the parameter `C_freq` or `H_freq`). Equivalently, the function `CT_Density_Plot()` can represent both density distributions or only one of them. Appendix B presents the empirical density probability functions for several experiments, where sample sizes vary from $N = (4, 4)$ to $N = (5, 5, 4, 4, 4, 4, 4)$.

Example 2 (Cont.) *Graphical visualization of the probability distributions and the density distributions of Example 2 generated by simulation. The first row of Figure 2 compares the probability distribution of the Concordance coefficient and the Kruskal-Wallis statistic. The second row of Figure 2 shows the probability density function of the Concordance coefficient (continuous line) and the Kruskal-Wallis statistic (dashed line). Note that the H statistic has been normalized between 0 and 1.*

```
set.seed(12)
Sample_Sizes <- c(10, 5, 3)
ProbDistr <- CT_Distribution(Sample_Sizes, Num_Sim = 25000, H = 1)
layout(matrix(c(1, 3, 2, 3), ncol=2))
CT_Probability_Plot(C_freq = ProbDistr$C_freq, H_freq = ProbDistr$H_freq)
CT_Density_Plot(C_freq = ProbDistr$C_freq, H_freq = ProbDistr$H_freq)
```

As we mentioned in Figure 1, Figure 2 also shows that similar values of the Kruskal-Wallis statistic present very different probabilities, and this leads to a less smooth function than that presented by the Concordance coefficient. We can also see that the Concordance coefficient presents a more symmetrical distribution. This performance is generalized and, therefore, we consider that the Concordance coefficient is more reliable than the Kruskal-Wallis statistic.

The **ConcordanceTest** package also contains the function `LOP()`, which solves the Linear Ordering Problem from a square data matrix. This function allows to calculate the disorder of a permutation of elements from the preference matrix induced by that permutation and, therefore, it is necessary for the calculation of the Concordance coefficient. The function `LOP()` is used by functions `CT_Distribution()`, `CT_Hypothesis_Test()` and `CT_Coefficient()`. It is used as follows:

```
LOP(mat_LOP)
```

where `mat_LOP` is the preference matrix defining the Linear Ordering Problem, a numeric square matrix for which we want to obtain the permutation of rows/columns that maximizes the sum of the elements above the main diagonal.

The function `LOP()` returns a list with the following elements: `obj_val` is the optimal value of the solution of the Linear Ordering Problem, that is, the sum of the elements above the main diagonal

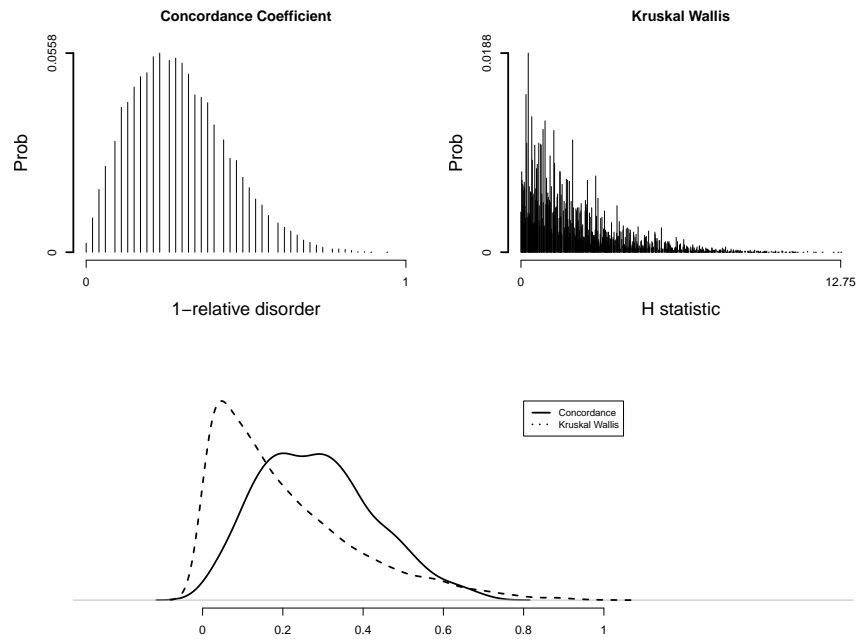


Figure 2: Probability distributions (first row) and density distributions (second row) of the Concordance coefficient ($\tau_c=1$ -relative disorder) and the Kruskal-Wallis statistic (H), with sample sizes $N = (10,5,3)$.

under the permutation rows/columns solution, permutation is the solution of the Linear Ordering Problem, that is, the rows/columns permutation, and `permutation_matrix` is the optimal permutation matrix of the Linear Ordering Problem.

Example 2 (Cont.) *The matrix of preferences between treatments observed in Example 2 was:*

$$\begin{array}{c} A \\ B \\ C \end{array} \begin{pmatrix} & A & B & C \\ A & - & 43 & 19 \\ B & 7 & - & 2 \\ C & 11 & 13 & - \end{pmatrix}$$

If we apply the function `LOP()` on this preference matrix we obtain the following results:

```
mat_LOP <- matrix(c(0,7,11,43,0,13,19,2,0), nrow=3)
LOP(mat_LOP)

$obj_val
[1] 75

$permutation
[1] 1 3 2

$permutation_matrix
  [,1] [,2] [,3]
[1,]  0   1   1
[2,]  0   0   0
[3,]  0   1   0
```

As we saw previously, the order between treatments that maximizes the order between patients corresponds to the order $(A C B)$ (`permutation = 1 3 2`), satisfying `obj_val = 75` of the preferences contained in the matrix.

5 Comparison with `kruskal.test()` function from `stats` package

The well-known `stats` package contains, among many other functions, the function `kruskal.test()` that performs a Kruskal-Wallis rank sum test. In this section, we compare the results obtained with the **ConcordanceTest** package presented in this work and the function `kruskal.test()`, making use of the dataset from [Hollander and Wolfe \(1973\)](#) referenced in the `kruskal.test()` examples.

Example 4 Comparison of `kruskal.test()` (`stats` package) and `CT_Hypothesis_Test()` functions with 25,000 simulations (**ConcordanceTest** package) using the dataset from [Hollander and Wolfe \(1973\)](#).

```
## Hollander & Wolfe (1973), 116.
## Mucociliary efficiency from the rate of removal of dust in normal
## subjects, subjects with obstructive airway disease, and subjects
## with asbestosis.

x <- c(2.9, 3.0, 2.5, 2.6, 3.2) # normal subjects
y <- c(3.8, 2.7, 4.0, 2.4)     # with obstructive airway disease
z <- c(2.8, 3.4, 3.7, 2.2, 2.0) # with asbestosis
Sample_List <- list(x, y, z)

kruskal.test(Sample_List)

      Kruskal-Wallis rank sum test

data: Sample_List
Kruskal-Wallis chi-squared = 0.77143, df = 2, p-value = 0.68

set.seed(12)
CT_Hypothesis_Test(Sample_List, Num_Sim = 25000, H = 1)

results
      Statistic p-value
Concordance coefficient    0.188 0.78408
Kruskal-Wallis            0.771 0.71080

$C_p_value
[1] 0.78408

$H_p_value
[1] 0.7108
```

As can be observed, the value of the Kruskal-Wallis statistic is the same with both functions (0.771). However, the p-values associated with the statistic differ.

The Kruskal-Wallis statistic follows approximately a χ^2 distribution with degrees of freedom equal to the number of groups minus 1 ([Kruskal and Wallis, 1952](#)). For this reason, the function `kruskal.test()` uses a χ^2 distribution to approximate the p-value (using the function `pchisq()`). In the case of the function `CT_Hypothesis_Test()`, it calculates the p-values using the simulations performed (25,000 in this example).

The function `CT_Distribution()` of the **ConcordanceTest** package allows the probability distribution tables of the Concordance coefficient and Kruskal-Wallis statistic to be computed, and they can be obtained exactly or by simulation. We can get the exact probability distribution tables and, consequently, the exact p-values in Example 4 with

```
CT_Distribution(c(5,4,5), Num_Sim = 0, H = 1)
```

In Example 4, the exact p-value for the Kruskal-Wallis statistic is 0.71077. Therefore, the difference between our p-value obtained with 25,000 simulations (0.71080) and the exact one is 0.00003, while the difference between the p-value approximated by the χ^2 distribution (0.68) and the exact one is 0.03077. Regarding the Concordance coefficient, the exact p-value is 0.78468, hence, the difference between our p-value obtained with 25,000 simulations (0.78408) and the exact one is 0.0006.

It is worth noting that the function `kruskal.test()` uses the χ^2 distribution to approximate the p-value regardless of the size of the samples, but [Kruskal and Wallis \(1952\)](#) state that the Kruskal-Wallis statistic is distributed approximately as a χ^2 , unless the samples are too small, in which case special

approximations or exact tables should be provided. On the contrary, the **ConcordanceTest** package can always obtain a good approximation of the p-values, regardless of the size of the samples, as long as a high number of simulations is used.

6 Final remarks and future research

A new measure based on the Kendall- τ distance is presented in this work to estimate the *concordance* of different samples. A statistical test to determine when different observations come from the same distribution is also introduced. A comparison with the classical Kruskal-Wallis test is introduced to show that both tests differ. As we have shown, the proposed coefficient is more appropriate and reliable than rank-based methods. This work also describes the R package **ConcordanceTest** (Alcaraz et al., 2022), which contains all the functions needed to work with the proposed Concordance coefficient and allows its comparison with the Kruskal-Wallis statistic.

This work aims to be an introduction of the new concordance measure between samples, but there still remains much to be done. There is a new problem and further challenges for researchers, for example: studying the asymptotic distribution of the Concordance coefficient, exploring the possibility of finding the exact distribution with the help of modern computing, or analyzing the power of the Concordance test presented in this work, among others.

7 Acknowledgments

The authors thank the grants PID2019-105952GB-I00 funded by Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación /10.13039/501100011033, Spain, and PROMETEO/2021/063 funded by the government of the Valencian Community, Spain.

Bibliography

- J. Alcaraz, L. Anton-Sanchez, and J. F. Monge. *ConcordanceTest: An Alternative to the Kruskal-Wallis Based on the Kendall Tau Distance*, 2022. URL <https://CRAN.R-project.org/package=ConcordanceTest>. R package version 1.0.2. [p27, 39]
- A. Alhakim and W. Hooper. A non-parametric test for several independent samples. *Journal of Nonparametric Statistics*, 20:253–261, 2008. URL <https://doi.org/10.1080/10485250801976741>. [p26]
- J. Aparicio, M. Landete, and J. F. Monge. A linear ordering problem of sets. *Annals of Operations Research*, 288(1):45–64, 2020. URL <https://doi.org/10.1007/s10479-019-03473-y>. [p26, 28, 29, 30]
- K. J. Arrow. *Social Choice and Individual Values*. Wiley, New York, 1951. [p29]
- J. F. Box. R. A. Fisher and the design of experiments, 1922-1926. *The American Statistician*, 34:1–7, 1980. URL <https://doi.org/10.2307/2682986>. [p32]
- R. Fisher. *The Design of Experiments*. Hafner Publishing Company, 1st edition, 1935. [p26, 32]
- M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11:86–92, 1940. URL <https://doi.org/10.1214/aoms/1177731944>. [p26]
- M. Hollander and D. Wolfe. *Nonparametric Statistical Methods*. Wiley series in probability and mathematical statistics. John Wiley & Sons, New York, 1973. URL <https://doi.org/10.1002/bimj.19750170808>. [p38]
- E. Irurozki, B. Calvo, and J. Lozano. Permallows: An R Package for Mallows and Generalized Mallows Models. *Journal of Statistical Software*, 71(12):1–30, 2016. URL <https://doi.org/10.18637/jss.v071.i12>. [p27]
- J. Kemeny. Mathematics without numbers. *Daedalus*, 88:577–591, 1959. [p29]
- M. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938. URL <https://doi.org/10.2307/2332226>. [p26, 27]

- M. Kendall and B. B. Smith. The problem of m ranking. *The Annals of Mathematical Statistics*, 10:275–287, 1939. URL <https://doi.org/10.1214/aoms/1177732186>. [p26]
- W. H. Kruskal. Ordinal measures of association. *Journal of the American Statistical Association*, 53: 814–861, 1958. URL <https://doi.org/10.1080/01621459.1958.10501481>. [p26]
- W. H. Kruskal and W. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47:583–618, 1952. URL <https://doi.org/10.1080/01621459.1952.10483441>. [p26, 27, 38]
- H. Mann and D. Whitney. On a test of whether one or two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60, 1947. URL <https://doi.org/10.1214/aoms/1177730491>. [p26, 27]
- R. Martí and G. Reinelt. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinational Optimization*. Springer, 1st edition, 2011. URL <https://doi.org/10.1007/978-3-642-16729-4>. [p29]
- A. McLeod. *Kendall: Kendall Rank Correlation and Mann-Kendall Trend Test*, 2011. URL <https://CRAN.R-project.org/package=Kendall>. R package version 2.2. [p27]
- J. P. Meyer and M. A. Seaman. *Kruskal-Wallis Exact Probability Tables*. 2015. URL <https://web.archive.org/web/20181017173535/http://faculty.virginia.edu/kruskal-wallis/>. [p33]
- E. Pitman. Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society*, 4:119–130, 1937. URL <https://doi.org/10.2307/2984124>. [p26]
- Z. Qian and P. Yu. Weighted distance-based models for ranking data using the R package rankdist. *Journal of Statistical Software*, 90(5):1–31, 2019. URL <https://doi.org/10.18637/jss.v090.i05>. [p27]
- P. Savicky. *pspearman: Spearman's Rank Correlation Test*, 2014. URL <https://CRAN.R-project.org/package=pspearman>. R package version 0.3-0. [p27]
- O. Sorensen, M. Crispino, Q. Liu, and V. Vitelli. BayesMallows: An R Package for the Bayesian Mallows Model. *The R Journal*, 12(1):324–342, June 2020. URL <https://doi.org/10.32614/RJ-2020-026>. [p27]
- C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:72–101, 1904. URL <https://doi.org/10.2307/1412159>. [p26]
- J. D. Spurrier. On the null distribution of the Kruskal-Wallis statistic. *Nonparametric Statistics*, 15: 695–691, 2003. URL <https://doi.org/10.1080/10485250310001634719>. [p28]
- H. Stern. Models for distributions on permutations. *Journal of the American Statistical Association*, 85: 558–564, 1990. URL <https://doi.org/10.1080/01621459.1990.10476235>. [p32]
- J. Terpstra and R. Magel. A new nonparametric test for the ordered alternative problem. *Nonparametric Statistics*, 15:289–301, 2003. URL <https://doi.org/10.1080/1048525031000078349>. [p26]
- W. J. Welch. Construction of permutation tests. *Journal of the American Statistical Association*, 85:693–698, 1990. URL <https://doi.org/10.1080/01621459.1990.10474929>. [p32]
- F. Wilcoxon. Individual comparisons by ranking method. *Biometrics*, 1:80–83, 1945. URL <https://doi.org/10.2307/3001968>. [p26]
- M. Zahid and H. Swart. The borda majority count. *Information Sciences*, 295:429–440, 2015. URL <https://doi.org/10.1016/j.ins.2014.10.044>. [p29]

1 Appendix A: Results in the experiment with sample sizes $N = (2, 2, 2)$

Table 6 shows the Concordance coefficient (τ_c) and Kruskal-Wallis statistic (H) for all possible results in an experiment with three treatments and two people in each treatment.

<i>dis</i>	τ_c	<i>H</i>	<i>dis</i>	τ_c	<i>H</i>	<i>dis</i>	τ_c	<i>H</i>
a a b b c c	0	1.0000 4.57	b a a b c c	2	0.6667 3.43	c a a b b c	4	0.3333 1.14
a a b c b c	1	0.8333 3.71	b a a c b c	3	0.5000 2.00	c a a b c b	3	0.5000 2.00
a a b c c b	2	0.6667 3.43	b a a c c b	4	0.3333 1.14	c a a c b b	2	0.6667 3.43
a a c b b c	2	0.6667 3.43	b a b a c c	1	0.8333 3.71	c a b a b c	5	0.1667 0.29
a a c b c b	1	0.8333 3.71	b a b c a c	2	0.6667 2.57	c a b a c b	4	0.3333 0.86
a a c c b b	0	1.0000 4.57	b a b c c a	3	0.5000 2.00	c a b b a c	6	0.0000 0.00
a b a b c c	1	0.8333 3.71	b a c a b c	4	0.3333 0.86	c a b b c a	5	0.1667 0.29
a b a c b c	2	0.6667 2.57	b a c a c b	5	0.1667 0.29	c a b c a b	3	0.5000 1.14
a b a c c b	3	0.5000 2.00	b a c b a c	3	0.5000 1.14	c a b c b a	4	0.3333 0.86
a b b a c c	2	0.6667 3.43	b a c b c a	4	0.3333 0.86	c a c a b b	1	0.8333 3.71
a b b c a c	3	0.5000 2.00	b a c c a b	6	0.0000 0.00	c a c b a b	2	0.6667 2.57
a b b c c a	4	0.3333 1.14	b a c c b a	5	0.1667 0.29	c a c b b a	3	0.5000 2.00
a b c a b c	3	0.5000 1.14	b b a a c c	0	1.0000 4.57	c b a a b c	6	0.0000 0.00
a b c a c b	4	0.3333 0.86	b b a c a c	1	0.8333 3.71	c b a a c b	5	0.1667 0.29
a b c b a c	4	0.3333 0.86	b b a c c a	2	0.6667 3.43	c b a b a c	5	0.1667 0.29
a b c b c a	5	0.1667 0.29	b b c a a c	2	0.6667 3.43	c b a b c a	4	0.3333 0.86
a b c c a b	5	0.1667 0.29	b b c a c a	1	0.8333 3.71	c b a c a b	4	0.3333 0.86
a b c c b a	6	0.0000 0.00	b b c c a a	0	1.0000 4.57	c b a c b a	3	0.5000 1.14
a c a b b c	3	0.5000 2.00	b c a a b c	5	0.1667 0.29	c b b a a c	4	0.3333 1.14
a c a b c b	2	0.6667 2.57	b c a a c b	6	0.0000 0.00	c b b a c a	3	0.5000 2.00
a c a c b b	1	0.8333 3.71	b c a b a c	4	0.3333 0.86	c b b c a a	2	0.6667 3.43
a c b a b c	4	0.3333 0.86	b c a b c a	3	0.5000 1.14	c b c a a b	3	0.5000 2.00
a c b a c b	3	0.5000 1.14	b c a c a b	5	0.1667 0.29	c b c a b a	2	0.6667 2.57
a c b b a c	5	0.1667 0.29	b c a c b a	4	0.3333 0.86	c b c b a a	1	0.8333 3.71
a c b b c a	6	0.0000 0.00	b c b a a c	3	0.5000 2.00	c c a a b b	0	1.0000 4.57
a c b c a b	4	0.3333 0.86	b c b a c a	2	0.6667 2.57	c c a b a b	1	0.8333 3.71
a c b c b a	5	0.1667 0.29	b c b c a a	1	0.8333 3.71	c c a b b a	2	0.6667 3.43
a c c a b b	2	0.6667 3.43	b c c a a b	4	0.3333 1.14	c c b a a b	2	0.6667 3.43
a c c b a b	3	0.5000 2.00	b c c a b a	3	0.5000 2.00	c c b a b a	1	0.8333 3.71
a c c b b a	4	0.3333 1.14	b c c b a a	2	0.6667 3.43	c c b b a a	0	1.0000 4.57

Table 6: Concordance coefficient (τ_c) and Kruskal-Wallis statistic (H) for all possible results in an experiment with sample sizes $N = (2, 2, 2)$.

2 Appendix B: Comparison of distributions

Table 7 shows the probability density function of the Concordance coefficient (continuous lines) and the Kruskal-Wallis statistic (dashed lines) generated by simulation. Number of simulations 100,000. Note that the H statistic has been normalized between 0 and 1.

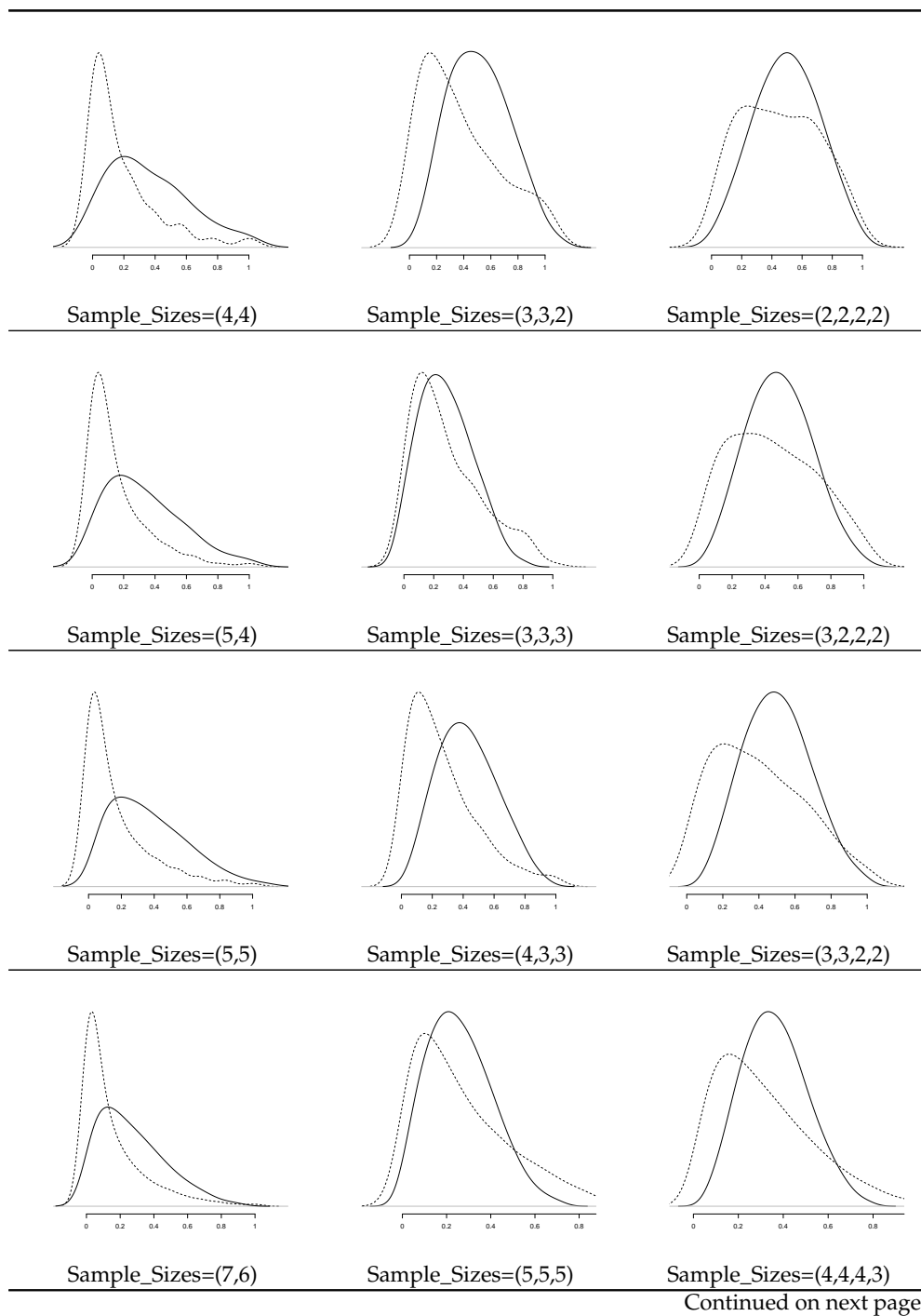


Table 7: Empirical density probability functions for several experiments (Concordance coefficient in continuous lines and Kruskal-Wallis statistic in dashed lines), where sample sizes vary from $N = (4, 4)$ to $N = (5, 5, 4, 4, 4, 4, 4)$.

Table 7 – continued from previous page

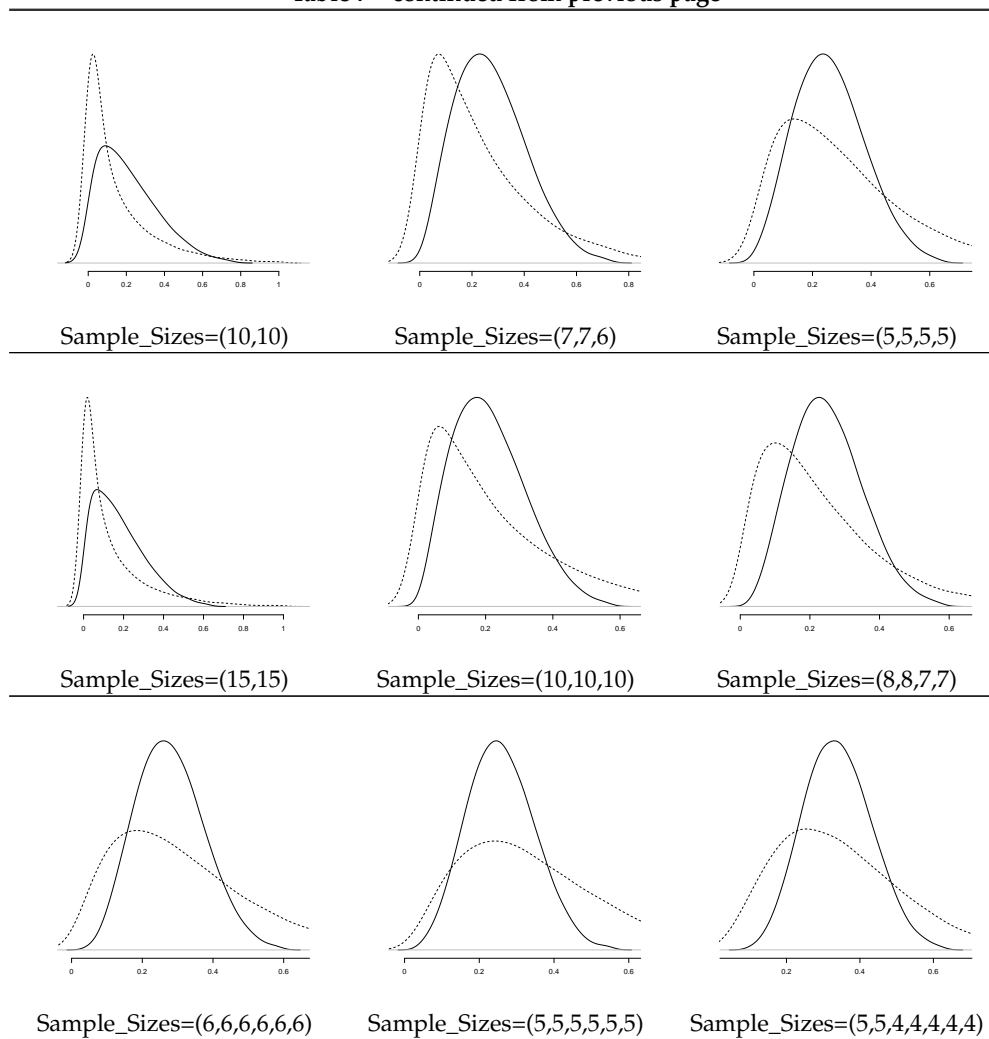


Table 7: Empirical density probability functions for several experiments (Concordance coefficient in continuous lines and Kruskal-Wallis statistic in dashed lines), where sample sizes vary from $N = (4,4)$ to $N = (5,5,4,4,4,4,4)$.

3 Appendix C: Concordance coefficient p-values

In order to compute the probability distribution of the Concordance coefficient, the enumeration of all the permutations of elements from an order is required. Note for example that if we have 4 samples with 6 elements each, $N = (6, 6, 6, 6)$, the number of possible results in the experiment is $24!/6!6!6!6! = 2.15433 \cdot 10^{20}$. The total computational time to compute the Concordance coefficient for all $2.15433 \cdot 10^{20}$ possibilities was more than 60 days in an Intel(R) Xeon (R) processor CPU E5-2650 v3 @ 2.30 GHz, 20 cores and RAM 64 GiB. Algorithm 1 presents the recursive function used to evaluate the Concordance coefficient probability distribution.

Algorithm 1: Algorithm to compute the exact probability distribution function of the Concordance coefficient τ_c

```

Data:
  p : ordered array of integers with ties.
  n : length of p.
  Frequency[0, max(disorder)].
1 Main (p)
2   Permutation(p,0,n);
3   return
4 Permutation (p,s,n):
5   Frequency[Disorder(p)]++;
6   int tmp = 0;
7   if s < n then
8     for i = n - 2 : i ≥ s; i - - do
9       for j = i + 1; j < n; j + + do
10        if p[i] ≠ p[j] then
11          [ tmp = p[i]; p[i] = p[j]; p[j] = tmp; Permutation(p,i + 1,n);
12          tmp=p[i];
13          for j = i + 1; j < n; j + + do
14            [ p[k] = p[k + +];
15            [ p[n - 1] = tmp;
16        return
17 Disorder (p)
18   /* Evaluate the disorder and the Concordance coefficient of permutation p */
19   return

```

Tables 8, 9 and 10 show the critical values and exact p-values of the Concordance coefficient τ_c at desired significance levels of 0.10, 0.05 and 0.01 for $k=2$, $k=3$ and $k=4$ samples, respectively.

Sample Sizes	.10			.05			.01		
	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value
4 1									
4 2									
4 3	0	1.000000	0.057143						
4 4	1	0.875000	0.057143	0	1.000000	0.028571			
5 1									
5 2	0	1.000000	0.095238						
5 3	1	0.857143	0.071429	0	1.000000	0.035714			
5 4	2	0.800000	0.063492	1	0.900000	0.031746			
5 5	4	0.666667	0.095238	2	0.833333	0.031746	0	1.000000	0.007937
6 1									
6 2	0	1.000000	0.071429						
6 3	2	0.777778	0.095238	1	0.888889	0.047619			
6 4	3	0.750000	0.066667	2	0.833333	0.038095	0	1.000000	0.009524
6 5	5	0.666667	0.082251	3	0.800000	0.030303	1	0.933333	0.008658
6 6	7	0.611111	0.093074	5	0.722222	0.041126	2	0.888889	0.008658
7 1									
7 2	0	1.000000	0.055556						
7 3	2	0.800000	0.066667	1	0.900000	0.033333			
7 4	4	0.714286	0.072727	3	0.785714	0.042424	0	1.000000	0.006061
7 5	6	0.647059	0.073232	5	0.705882	0.047980	1	0.941176	0.005051
7 6	8	0.619048	0.073427	6	0.714286	0.034965	3	0.857143	0.008159
7 7	11	0.541667	0.097319	8	0.666667	0.037879	4	0.833333	0.006993
8 1									
8 2	1	0.875000	0.088889	0	1.000000	0.044444			
8 3	3	0.750000	0.084848	2	0.833333	0.048485			
8 4	5	0.687500	0.072727	4	0.750000	0.048485	1	0.937500	0.008081
8 5	8	0.600000	0.093240	6	0.700000	0.045066	2	0.900000	0.006216
8 6	10	0.583333	0.081252	8	0.666667	0.042624	4	0.833333	0.007992
8 7	13	0.535714	0.093862	10	0.642857	0.040093	6	0.785714	0.009324
8 8	15	0.531250	0.082984	13	0.593750	0.049883	7	0.781250	0.006993
9 1									
9 2	1	0.888889	0.072727	0	1.000000	0.036364			
9 3	3	0.769231	0.063636	2	0.846154	0.036364	0	1.000000	0.009091
9 4	6	0.666667	0.075524	4	0.777778	0.033566	1	0.944444	0.005594
9 5	9	0.590909	0.082917	7	0.681818	0.041958	3	0.863636	0.006993
9 6	12	0.555556	0.087912	10	0.629630	0.049550	5	0.814815	0.007592
9 7	15	0.516129	0.090734	12	0.612903	0.041783	7	0.774194	0.007867
9 8	18	0.500000	0.092719	15	0.583333	0.046401	9	0.750000	0.007898
9 9	21	0.475000	0.093912	17	0.575000	0.039984	11	0.725000	0.007775
10 1									
10 2	1	0.900000	0.060606	0	1.000000	0.030303			
10 3	4	0.733333	0.076923	3	0.800000	0.048951	0	1.000000	0.006993
10 4	7	0.650000	0.075924	5	0.750000	0.035964	2	0.900000	0.007992
10 5	11	0.560000	0.099234	8	0.680000	0.039960	4	0.840000	0.007992
10 6	14	0.533333	0.093407	11	0.633333	0.041958	6	0.800000	0.007493
10 7	17	0.514286	0.087824	14	0.600000	0.043089	9	0.742857	0.009667
10 8	20	0.500000	0.083139	17	0.575000	0.043421	11	0.725000	0.008547
10 9	24	0.466667	0.094720	20	0.555556	0.043474	13	0.711111	0.007621
10 10	27	0.460000	0.089210	23	0.540000	0.043257	16	0.680000	0.008931
11 1									
11 2	1	0.909091	0.051282	0	1.000000	0.025641			
11 3	5	0.687500	0.087912	3	0.812500	0.038462	0	1.000000	0.005495
11 4	8	0.636364	0.077656	6	0.727273	0.039560	2	0.909091	0.005861
11 5	12	0.555556	0.089744	9	0.666667	0.038004	5	0.814815	0.008700
11 6	16	0.515152	0.098255	13	0.606061	0.047673	7	0.787879	0.007111
11 7	19	0.500000	0.085344	16	0.578947	0.044118	10	0.736842	0.008296
11 8	23	0.477273	0.090842	19	0.568182	0.040883	13	0.704545	0.009103
11 9	27	0.448980	0.095177	23	0.530612	0.046452	16	0.673469	0.009693
11 10	31	0.436364	0.098618	26	0.527273	0.042964	18	0.672727	0.007950
11 11	34	0.433333	0.087946	30	0.500000	0.047307	21	0.650000	0.008330
12 1									
12 2	2	0.833333	0.087912	1	0.916667	0.043956			
12 3	5	0.722222	0.070330	4	0.777778	0.048352	1	0.944444	0.008791
12 4	9	0.625000	0.078022	7	0.708333	0.041758	3	0.875000	0.007692
12 5	13	0.566667	0.081771	11	0.633333	0.048481	6	0.800000	0.009373
12 6	17	0.527778	0.083064	14	0.611111	0.041478	9	0.750000	0.009696

Continued on next page

Table 8: Critical values and exact p-values of the Concordance coefficient τ_c for $k=2$ samples.

Table 8 – continued from previous page

Sample Sizes	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	
12	7	21	0.500000	0.083115	18	0.571429	0.044931	12	0.714286	0.009764
12	8	26	0.458333	0.097880	22	0.541667	0.047345	15	0.687500	0.009558
12	9	30	0.444444	0.095451	26	0.518519	0.049073	18	0.666667	0.009288
12	10	34	0.433333	0.093090	29	0.516667	0.042570	21	0.650000	0.008957
12	11	38	0.424242	0.090842	33	0.500000	0.043879	24	0.636364	0.008625
12	12	42	0.416667	0.088734	37	0.486111	0.044902	27	0.625000	0.008293
13	1									
13	2	2	0.846154	0.076190	1	0.923077	0.038095			
13	3	6	0.684211	0.082143	4	0.789474	0.039286	1	0.947368	0.007143
13	4	10	0.615385	0.078992	8	0.692308	0.044538	3	0.884615	0.005882
13	5	15	0.531250	0.094538	12	0.625000	0.045985	7	0.781250	0.009804
13	6	19	0.512820	0.087424	16	0.589744	0.046218	10	0.743590	0.009214
13	7	24	0.466667	0.096801	20	0.555556	0.045562	13	0.711111	0.008462
13	8	28	0.461538	0.089046	24	0.538462	0.044553	17	0.673077	0.009937
13	9	33	0.431035	0.095557	28	0.517241	0.043376	20	0.655172	0.008910
13	10	37	0.430769	0.088294	33	0.492308	0.049329	24	0.630769	0.009888
13	11	42	0.408451	0.093307	37	0.478873	0.047448	27	0.619718	0.008848
13	12	47	0.397436	0.097642	41	0.474359	0.045711	31	0.602564	0.009556
13	13	51	0.392857	0.090847	45	0.464286	0.044117	34	0.595238	0.008601
14	1									
14	2	2	0.857143	0.066667	1	0.928571	0.033333			
14	3	7	0.666667	0.091176	5	0.761905	0.047059	1	0.952381	0.005882
14	4	11	0.607143	0.079085	9	0.678571	0.046405	4	0.857143	0.007843
14	5	16	0.542857	0.087031	13	0.628571	0.043688	7	0.800000	0.007224
14	6	21	0.500000	0.091331	17	0.595238	0.040764	11	0.738095	0.008720
14	7	26	0.469388	0.093774	22	0.551020	0.046096	15	0.693878	0.009684
14	8	31	0.446429	0.095018	26	0.535714	0.042149	18	0.678571	0.008125
14	9	36	0.428571	0.095574	31	0.507936	0.045585	22	0.650794	0.008568
14	10	41	0.414286	0.095643	36	0.485714	0.048404	26	0.628571	0.008851
14	11	46	0.402597	0.095427	40	0.480519	0.044228	30	0.610390	0.009022
14	12	51	0.392857	0.095012	45	0.464286	0.046354	34	0.595238	0.009114
14	13	56	0.384615	0.094479	50	0.450549	0.048173	38	0.582418	0.009150
14	14	61	0.377551	0.093868	55	0.438776	0.049736	42	0.571429	0.009146
15	1									
15	2	3	0.800000	0.088235	1	0.933333	0.029412			
15	3	7	0.681818	0.075980	5	0.772727	0.039216	2	0.909091	0.009804
15	4	12	0.600000	0.079979	10	0.666667	0.048504	5	0.833333	0.009288
15	5	18	0.513514	0.098297	14	0.621622	0.041796	8	0.783784	0.007740
15	6	23	0.488889	0.094833	19	0.577778	0.044855	12	0.733333	0.008367
15	7	28	0.461538	0.091085	24	0.538462	0.046522	16	0.692308	0.008526
15	8	33	0.450000	0.087332	29	0.516667	0.047304	20	0.666667	0.008456
15	9	39	0.417910	0.095507	34	0.492537	0.047584	24	0.641791	0.008255
15	10	44	0.413333	0.090971	39	0.480000	0.047524	29	0.613333	0.009616
15	11	50	0.390244	0.097262	44	0.463415	0.047262	33	0.597561	0.009154
15	12	55	0.388889	0.092610	49	0.455556	0.046866	37	0.588889	0.008710
15	13	61	0.371134	0.097721	54	0.443299	0.046394	42	0.567010	0.009635
15	14	66	0.371429	0.093216	59	0.438095	0.045875	46	0.561905	0.009115
15	15	72	0.357143	0.097526	64	0.428571	0.045334	51	0.544643	0.009875
16	1									
16	2	3	0.812500	0.078431	1	0.937500	0.026144			
16	3	8	0.666667	0.084623	6	0.750000	0.047472	2	0.916667	0.008256
16	4	14	0.562500	0.099484	11	0.656250	0.049948	5	0.843750	0.007430
16	5	19	0.525000	0.091012	15	0.625000	0.040100	9	0.775000	0.008158
16	6	25	0.479167	0.098026	21	0.562500	0.048731	13	0.729167	0.007988
16	7	30	0.464286	0.088694	26	0.535714	0.046876	18	0.678571	0.009578
16	8	36	0.437500	0.092602	31	0.515625	0.044823	22	0.656250	0.008748
16	9	42	0.416667	0.095397	37	0.486111	0.049384	27	0.625000	0.009643
16	10	48	0.400000	0.097414	42	0.475000	0.046707	31	0.612500	0.008685
16	11	54	0.386364	0.098866	47	0.465909	0.044271	36	0.590909	0.009256
16	12	60	0.375000	0.099904	53	0.447917	0.047276	41	0.572917	0.009707
16	13	65	0.375000	0.091611	59	0.432692	0.049924	45	0.567308	0.008738
16	14	71	0.366071	0.092540	64	0.428571	0.047205	50	0.553571	0.009064
16	15	77	0.358333	0.093259	70	0.416667	0.049381	55	0.541667	0.009331
16	16	83	0.351562	0.093812	75	0.414062	0.046815	60	0.531250	0.009551
17	1									
17	2	3	0.823529	0.070175	2	0.882353	0.046784			
17	3	9	0.640000	0.092982	6	0.760000	0.040351	2	0.920000	0.007018

Continued on next page

Table 8: Critical values and exact p-values of the Concordance coefficient τ_c for $k=2$ samples.

Table 8 – continued from previous page

Sample Sizes	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value
17 4	15	0.558824	0.098580	11	0.676471	0.040434	6	0.823529	0.009023
17 5	20	0.523810	0.084909	17	0.595238	0.047695	10	0.761905	0.008582
17 6	26	0.490196	0.086501	22	0.568627	0.043766	15	0.705882	0.009867
17 7	33	0.440678	0.099490	28	0.525424	0.047171	19	0.677966	0.008518
17 8	39	0.426471	0.097491	34	0.500000	0.049474	24	0.647059	0.009005
17 9	45	0.407895	0.095246	39	0.486842	0.044566	29	0.618421	0.009248
17 10	51	0.400000	0.092922	45	0.470588	0.045937	34	0.600000	0.009341
17 11	57	0.387097	0.090623	51	0.451613	0.046916	39	0.580645	0.009331
17 12	64	0.372549	0.097270	57	0.441176	0.047604	44	0.568627	0.009257
17 13	70	0.363636	0.094466	63	0.427273	0.048075	49	0.554545	0.009141
17 14	77	0.352941	0.099995	69	0.420168	0.048385	54	0.546219	0.008999
17 15	83	0.346457	0.096996	75	0.409449	0.048571	60	0.527559	0.009973
17 16	89	0.345588	0.094235	81	0.404412	0.048664	65	0.522059	0.009731
17 17	96	0.333333	0.098687	87	0.395833	0.048686	70	0.513889	0.009494
18 1									
18 2	4	0.777778	0.094737	2	0.888889	0.042105			
18 3	9	0.666667	0.079699	7	0.740741	0.046617	2	0.925926	0.006015
18 4	16	0.555556	0.098154	12	0.666667	0.042379	6	0.833333	0.007382
18 5	22	0.511111	0.094327	18	0.600000	0.045707	11	0.755556	0.008916
18 6	28	0.481481	0.089527	24	0.555556	0.047193	16	0.703704	0.009421
18 7	35	0.444444	0.096701	30	0.523810	0.047418	21	0.666667	0.009445
18 8	41	0.430556	0.090496	36	0.500000	0.046988	26	0.638889	0.009233
18 9	48	0.407407	0.095074	42	0.481481	0.046198	31	0.617284	0.008893
18 10	55	0.388889	0.098664	48	0.466667	0.045221	37	0.588889	0.009955
18 11	61	0.383838	0.092197	55	0.444444	0.049392	42	0.575758	0.009388
18 12	68	0.370370	0.094866	61	0.435185	0.047865	47	0.564815	0.008851
18 13	75	0.358974	0.097070	67	0.427350	0.046401	53	0.547009	0.009505
18 14	82	0.349206	0.098905	74	0.412698	0.049436	58	0.539683	0.008925
18 15	88	0.348148	0.092994	80	0.407407	0.047795	64	0.525926	0.009432
18 16	95	0.340278	0.094552	86	0.402778	0.046272	70	0.513889	0.009880
18 17	102	0.333333	0.095895	93	0.392157	0.048652	75	0.509804	0.009265
18 18	109	0.327160	0.097059	99	0.388889	0.047085	81	0.500000	0.009631
19 1									
19 2	4	0.789474	0.085714	2	0.894737	0.038095	0	1.000000	0.009524
19 3	10	0.642857	0.087013	7	0.750000	0.040260	3	0.892857	0.009091
19 4	17	0.552632	0.097346	13	0.657895	0.043817	7	0.815789	0.008583
19 5	23	0.510638	0.088368	19	0.595745	0.043902	12	0.744681	0.009270
19 6	30	0.473684	0.092321	25	0.561404	0.042778	17	0.701754	0.009001
19 7	37	0.439394	0.094199	32	0.515152	0.047622	22	0.666667	0.008489
19 8	44	0.421053	0.094915	38	0.500000	0.044792	28	0.631579	0.009436
19 9	51	0.400000	0.094882	45	0.470588	0.047700	33	0.611765	0.008572
19 10	58	0.389474	0.094392	52	0.452632	0.049957	39	0.589474	0.009074
19 11	65	0.375000	0.093614	58	0.442308	0.046502	45	0.567308	0.009429
19 12	72	0.368421	0.092664	65	0.429825	0.048074	51	0.552632	0.009674
19 13	80	0.349594	0.099454	72	0.414634	0.049346	57	0.536585	0.009835
19 14	87	0.345865	0.097861	78	0.413534	0.046065	63	0.526316	0.009935
19 15	94	0.338028	0.096301	85	0.401408	0.047054	69	0.514085	0.009986
19 16	101	0.335526	0.094785	92	0.394737	0.047883	74	0.513158	0.009009
19 17	109	0.322981	0.099827	99	0.385093	0.048578	81	0.496894	0.009991
19 18	116	0.321637	0.098072	106	0.380117	0.049163	87	0.491228	0.009960
19 19	123	0.316667	0.096409	113	0.372222	0.049656	93	0.483333	0.009914
20 1	0	1.000000	0.095238						
20 2	4	0.800000	0.077922	2	0.900000	0.034632	0	1.000000	0.008658
20 3	11	0.633333	0.093732	8	0.733333	0.046302	3	0.900000	0.007905
20 4	18	0.550000	0.096932	14	0.650000	0.045360	8	0.800000	0.009976
20 5	25	0.500000	0.096970	20	0.600000	0.042349	13	0.740000	0.009561
20 6	32	0.466667	0.094905	27	0.550000	0.045858	18	0.700000	0.008652
20 7	39	0.442857	0.091932	34	0.514286	0.047798	24	0.657143	0.009315
20 8	47	0.412500	0.099062	41	0.487500	0.048749	30	0.625000	0.009617
20 9	54	0.400000	0.094682	48	0.466667	0.049091	36	0.600000	0.009687
20 10	62	0.380000	0.099577	55	0.450000	0.049031	42	0.580000	0.009616
20 11	69	0.372727	0.094896	62	0.436364	0.048718	48	0.563636	0.009458
20 12	77	0.358333	0.098543	69	0.425000	0.048240	54	0.550000	0.009249
20 13	84	0.353846	0.093978	76	0.415385	0.047661	60	0.538462	0.009012
20 14	92	0.342857	0.096865	83	0.407143	0.047021	67	0.521429	0.009796
20 15	100	0.333333	0.099377	90	0.400000	0.046348	73	0.513333	0.009462
20 16	107	0.331250	0.094950	98	0.387500	0.049370	79	0.506250	0.009140

Continued on next page

Table 8: Critical values and exact p-values of the Concordance coefficient τ_c for $k=2$ samples.

Table 8 – continued from previous page

Sample Sizes	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value
20 17	115	0.323529	0.097069	105	0.382353	0.048471	86	0.494118	0.009721
20 18	123	0.316667	0.098957	112	0.377778	0.047600	92	0.488889	0.009363
20 19	130	0.315789	0.094835	119	0.373684	0.046761	99	0.478947	0.009856
20 20	138	0.310000	0.096500	127	0.365000	0.049090	105	0.475000	0.009484

Table 8: Critical values and exact p-values of the Concordance coefficient τ_c for $k=2$ samples.

Sample Sizes	.10			.05			.01		
	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value
2 1 1									
2 2 1									
2 2 2	0	1.000000	0.066667						
3 1 1									
3 2 1									
3 2 2	1	0.875000	0.085714	0	1.000000	0.028571			
3 3 1	0	1.000000	0.042857	0	1.000000	0.042857			
3 3 2	2	0.800000	0.085714	1	0.900000	0.032143			
3 3 3	3	0.769231	0.064286	2	0.846154	0.028571	0	1.000000	0.003571
4 1 1									
4 2 1	0	1.000000	0.057143						
4 2 2	1	0.900000	0.042857	1	0.900000	0.042857			
4 3 1	1	0.888889	0.064286	0	1.000000	0.021429			
4 3 2	3	0.769231	0.077778	2	0.846154	0.038095	0	1.000000	0.004762
4 3 3	5	0.687500	0.090000	3	0.812500	0.025714	1	0.937500	0.004286
4 4 1	2	0.833333	0.060317	1	0.916667	0.028571	0	1.000000	0.009524
4 4 2	4	0.750000	0.060952	3	0.812500	0.032381	1	0.937500	0.005714
4 4 3	7	0.650000	0.095065	5	0.750000	0.035325	3	0.850000	0.009351
4 4 4	9	0.625000	0.086580	7	0.708333	0.036883	4	0.833333	0.006580
5 1 1									
5 2 1	1	0.875000	0.095238	0	1.000000	0.035714			
5 2 2	2	0.833333	0.058201	1	0.916667	0.023810	0	1.000000	0.007937
5 3 1	2	0.818182	0.075397	1	0.909091	0.035714			
5 3 2	4	0.733333	0.072222	3	0.800000	0.038889	1	0.933333	0.007143
5 3 3	6	0.684211	0.070130	5	0.736842	0.041558	2	0.894737	0.005195
5 4 1	4	0.714286	0.098413	2	0.857143	0.031746	0	1.000000	0.004762
5 4 2	6	0.684211	0.079654	5	0.736842	0.049062	2	0.894737	0.006926
5 4 3	9	0.608696	0.098341	7	0.695652	0.042641	4	0.826087	0.008009
5 4 4	11	0.607143	0.079343	9	0.678571	0.037163	6	0.785714	0.008658
5 5 1	5	0.705882	0.077201	4	0.764706	0.047619	1	0.941176	0.006494
5 5 2	8	0.636364	0.084416	6	0.727273	0.035714	3	0.863636	0.006133
5 5 3	11	0.592593	0.089022	9	0.666667	0.042374	5	0.814815	0.005828
5 5 4	14	0.562500	0.089498	12	0.625000	0.047072	8	0.750000	0.009039
5 5 5	17	0.540541	0.088887	14	0.621622	0.036630	10	0.729730	0.008016
6 1 1									
6 2 1	1	0.900000	0.063492	0	1.000000	0.023810			
6 2 2	3	0.785714	0.066667	2	0.857143	0.034921	0	1.000000	0.004762
6 3 1	3	0.769231	0.083333	2	0.846154	0.045238	0	1.000000	0.007143
6 3 2	5	0.722222	0.067532	4	0.777778	0.039394	1	0.944444	0.003896
6 3 3	8	0.636364	0.087554	6	0.727273	0.035390	3	0.863636	0.005844
6 4 1	5	0.705882	0.089177	3	0.823529	0.032035	1	0.941176	0.007792
6 4 2	8	0.636364	0.096537	6	0.727273	0.041414	3	0.863636	0.007359
6 4 3	11	0.592593	0.099933	9	0.666667	0.048119	5	0.814815	0.006893
6 4 4	14	0.562500	0.099310	11	0.656250	0.036934	7	0.781250	0.006394
6 5 1	7	0.650000	0.094156	5	0.750000	0.040043	2	0.900000	0.007576
6 5 2	10	0.615385	0.087468	8	0.692308	0.041570	4	0.846154	0.005661
6 5 3	13	0.580645	0.081205	11	0.645161	0.041625	7	0.774194	0.007635
6 5 4	17	0.540541	0.097296	14	0.621622	0.040721	10	0.729730	0.009238
6 5 5	20	0.523810	0.087370	17	0.595238	0.039446	12	0.714286	0.007222
6 6 1	9	0.625000	0.095571	7	0.708333	0.046287	3	0.875000	0.006660
6 6 2	12	0.600000	0.080039	10	0.666667	0.041173	6	0.800000	0.007588
6 6 3	16	0.555556	0.089258	13	0.638889	0.036473	9	0.750000	0.008044
6 6 4	20	0.523810	0.094960	17	0.595238	0.043424	12	0.714286	0.008249
6 6 5	24	0.500000	0.098268	21	0.562500	0.049106	15	0.687500	0.008321
6 6 6	27	0.500000	0.082204	24	0.555556	0.042636	18	0.666667	0.008323

Continued on next page

Table 9: Critical values and exact p-values of the Concordance coefficient τ_c for $k=3$ samples.

Table 9 – continued from previous page

Sample Sizes	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value		
7	1	1	0	1.000000	0.083333						
7	2	1	2	0.818182	0.094444	1	0.909091	0.044444			
7	2	2	4	0.750000	0.076768	3	0.812500	0.042424	1	0.937500	0.009091
7	3	1	4	0.733333	0.092424	2	0.866667	0.028788	0	1.000000	0.004545
7	3	2	7	0.650000	0.098737	5	0.750000	0.039394	2	0.900000	0.006061
7	3	3	9	0.640000	0.071270	8	0.680000	0.047669	4	0.840000	0.006294
7	4	1	6	0.684211	0.083333	4	0.789474	0.033333	1	0.947368	0.004545
7	4	2	9	0.640000	0.078788	7	0.720000	0.035664	4	0.840000	0.007692
7	4	3	12	0.600000	0.074026	10	0.666667	0.036680	6	0.800000	0.006061
7	4	4	16	0.555556	0.090541	13	0.638889	0.036572	9	0.750000	0.007779
7	5	1	8	0.652174	0.077506	6	0.739130	0.035354	3	0.869565	0.007770
7	5	2	12	0.586207	0.089494	10	0.655172	0.046481	6	0.793103	0.008658
7	5	3	16	0.542857	0.098957	13	0.628571	0.040904	9	0.742857	0.009108
7	5	4	19	0.536585	0.081531	17	0.585366	0.048012	12	0.707317	0.009257
7	5	5	23	0.510638	0.085929	20	0.574468	0.041698	15	0.680851	0.009291
7	6	1	11	0.592593	0.096820	8	0.703704	0.035881	5	0.814815	0.009907
7	6	2	15	0.558824	0.097303	12	0.647059	0.040593	8	0.764706	0.009135
7	6	3	19	0.525000	0.096083	16	0.600000	0.043746	11	0.725000	0.008244
7	6	4	23	0.510638	0.092608	20	0.574468	0.045458	14	0.702128	0.007419
7	6	5	27	0.490566	0.088632	24	0.547170	0.046362	18	0.660377	0.009231
7	6	6	31	0.483333	0.084562	28	0.533333	0.046592	21	0.650000	0.008248
7	7	1	13	0.580645	0.088462	11	0.645161	0.049728	6	0.806452	0.007653
7	7	2	17	0.552632	0.081371	15	0.605263	0.048067	10	0.736842	0.009368
7	7	3	22	0.511111	0.093660	19	0.577778	0.045940	13	0.711111	0.007505
7	7	4	26	0.500000	0.083679	23	0.557692	0.043172	17	0.673077	0.008389
7	7	5	31	0.474576	0.090678	27	0.542373	0.040628	21	0.644068	0.009101
7	7	6	36	0.454545	0.095828	32	0.515152	0.046525	25	0.621212	0.009656
7	7	7	40	0.452055	0.085655	36	0.506849	0.043267	28	0.616438	0.007945
8	1	1	0	1.000000	0.066667						
8	2	1	2	0.846154	0.068687	1	0.923077	0.032323			
8	2	2	5	0.722222	0.083502	3	0.833333	0.028283	1	0.944444	0.006061
8	3	1	5	0.705882	0.097980	3	0.823529	0.036364	1	0.941176	0.009091
8	3	2	8	0.652174	0.091064	6	0.739130	0.039627	3	0.869565	0.007615
8	3	3	11	0.607143	0.084582	9	0.678571	0.041026	5	0.821429	0.006394
8	4	1	7	0.681818	0.077389	5	0.772727	0.033877	2	0.909091	0.006216
8	4	2	11	0.607143	0.091553	9	0.678571	0.046309	5	0.821429	0.007681
8	4	3	14	0.588235	0.076546	12	0.647059	0.040884	8	0.764706	0.008560
8	4	4	18	0.550000	0.083417	16	0.600000	0.048629	11	0.725000	0.008991
8	5	1	10	0.615385	0.089355	8	0.692308	0.045732	4	0.846154	0.007881
8	5	2	14	0.575758	0.090768	11	0.666667	0.036408	7	0.787879	0.007489
8	5	3	18	0.538462	0.090415	15	0.615385	0.040041	10	0.743590	0.006990
8	5	4	22	0.521739	0.087620	19	0.586957	0.042130	14	0.695652	0.009212
8	5	5	26	0.500000	0.084260	23	0.557692	0.043404	17	0.673077	0.008280
8	6	1	13	0.580645	0.096881	10	0.677419	0.040004	6	0.806452	0.008614
8	6	2	17	0.552632	0.089066	14	0.631579	0.039832	9	0.763158	0.007065
8	6	3	21	0.533333	0.081197	18	0.600000	0.038672	13	0.711111	0.008335
8	6	4	26	0.500000	0.090348	23	0.557692	0.046990	17	0.673077	0.009265
8	6	5	31	0.474576	0.097034	27	0.542373	0.043907	21	0.644068	0.009983
8	6	6	35	0.469697	0.086285	32	0.515152	0.049960	24	0.636364	0.008137
8	7	1	15	0.571429	0.081138	13	0.628571	0.047786	8	0.771429	0.009091
8	7	2	20	0.534884	0.087307	17	0.604651	0.042356	12	0.720930	0.009450
8	7	3	25	0.500000	0.091071	22	0.560000	0.047473	16	0.680000	0.009437
8	7	4	30	0.482759	0.092169	26	0.551724	0.041134	20	0.655172	0.009202
8	7	5	35	0.461538	0.091936	31	0.523077	0.044101	24	0.630769	0.008929
8	7	6	40	0.452055	0.090824	36	0.506849	0.046234	28	0.616438	0.008639
8	7	7	45	0.437500	0.089477	41	0.487500	0.047863	32	0.600000	0.008348
8	8	1	18	0.550000	0.086668	15	0.625000	0.042022	10	0.750000	0.009297
8	8	2	23	0.520833	0.085381	20	0.583333	0.044213	14	0.708333	0.008570
8	8	3	29	0.482143	0.099335	25	0.553571	0.044954	18	0.678571	0.007749
8	8	4	34	0.468750	0.093356	30	0.531250	0.044695	23	0.640625	0.009074
8	8	5	39	0.458333	0.087277	35	0.513889	0.044004	27	0.625000	0.008054
8	8	6	45	0.437500	0.094509	40	0.500000	0.043013	32	0.600000	0.009033
8	8	7	50	0.431818	0.087903	46	0.477273	0.049055	37	0.579545	0.009894
8	8	8	56	0.416667	0.093322	51	0.468750	0.047287	41	0.572917	0.008809
9	1	1	0	1.000000	0.054545						
9	2	1	3	0.785714	0.093939	1	0.928571	0.024242	0	1.000000	0.009091
9	2	2	6	0.700000	0.090443	4	0.800000	0.035431	1	0.950000	0.004196

Continued on next page

Table 9: Critical values and exact p-values of the Concordance coefficient τ_c for $k=3$ samples.

Table 9 – continued from previous page

Sample Sizes	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value		
9	3	1	5	0.736842	0.069231	4	0.789474	0.044056	1	0.947368	0.006294
9	3	2	9	0.640000	0.084815	7	0.720000	0.039461	4	0.840000	0.009091
9	3	3	13	0.580645	0.096883	10	0.677419	0.036064	6	0.806452	0.006533
9	4	1	8	0.666667	0.073327	6	0.750000	0.034565	3	0.875000	0.007592
9	4	2	12	0.612903	0.077416	10	0.677419	0.040573	6	0.806452	0.007752
9	4	3	16	0.567568	0.078701	14	0.621622	0.044560	9	0.756757	0.007438
9	4	4	21	0.522727	0.097718	18	0.590909	0.046871	12	0.727273	0.007022
9	5	1	11	0.620690	0.075658	9	0.689655	0.040160	5	0.827586	0.007925
9	5	2	16	0.555556	0.091767	13	0.638889	0.040152	8	0.777778	0.006618
9	5	3	20	0.534884	0.083722	17	0.604651	0.039249	12	0.720930	0.008063
9	5	4	25	0.500000	0.092920	22	0.560000	0.047915	16	0.680000	0.009130
9	5	5	30	0.473684	0.099791	26	0.543860	0.044813	20	0.649123	0.009980
9	6	1	15	0.558824	0.097278	12	0.647059	0.043681	7	0.794118	0.007617
9	6	2	19	0.547619	0.082476	16	0.619048	0.039079	11	0.738095	0.008189
9	6	3	24	0.510204	0.086697	21	0.571429	0.044388	15	0.693878	0.008372
9	6	4	29	0.491228	0.088247	26	0.543860	0.048185	19	0.666667	0.008313
9	6	5	34	0.468750	0.088358	30	0.531250	0.041791	23	0.640625	0.008178
9	6	6	39	0.458333	0.087572	35	0.513889	0.044093	27	0.625000	0.008005
9	7	1	18	0.538462	0.094755	15	0.615385	0.046318	9	0.769231	0.007240
9	7	2	23	0.510638	0.092339	20	0.574468	0.048129	14	0.702128	0.009452
9	7	3	28	0.490909	0.088931	25	0.545455	0.048734	18	0.672727	0.008511
9	7	4	34	0.460317	0.099728	30	0.523810	0.048091	23	0.634921	0.009883
9	7	5	39	0.450704	0.092938	35	0.507042	0.047167	27	0.619718	0.008744
9	7	6	45	0.430380	0.099994	40	0.493671	0.045852	32	0.594937	0.009751
9	7	7	50	0.425287	0.092778	45	0.482759	0.044526	36	0.586207	0.008675
9	8	1	21	0.522727	0.091613	18	0.590909	0.047877	12	0.727273	0.009402
9	8	2	26	0.509434	0.083606	23	0.566038	0.045645	16	0.698113	0.007832
9	8	3	32	0.475410	0.090200	28	0.540984	0.042843	21	0.655738	0.008515
9	8	4	38	0.457143	0.094142	34	0.514286	0.047762	26	0.628571	0.008914
9	8	5	44	0.435897	0.096461	39	0.500000	0.043813	31	0.602564	0.009177
9	8	6	50	0.425287	0.097552	45	0.482759	0.047178	36	0.586207	0.009342
9	8	7	56	0.410526	0.098037	51	0.463158	0.049980	41	0.568421	0.009437
9	8	8	62	0.403846	0.098020	56	0.461538	0.045624	46	0.557692	0.009473
9	9	1	24	0.510204	0.089203	21	0.571429	0.049174	14	0.714286	0.008606
9	9	2	30	0.482759	0.091114	26	0.551724	0.043471	19	0.672414	0.008661
9	9	3	36	0.462687	0.091244	32	0.522388	0.046073	24	0.641791	0.008468
9	9	4	42	0.447368	0.089294	38	0.500000	0.047313	29	0.618421	0.008104
9	9	5	49	0.423529	0.099468	44	0.482353	0.048059	35	0.588235	0.009520
9	9	6	55	0.414894	0.095225	50	0.468085	0.048205	40	0.574468	0.008960
9	9	7	61	0.407767	0.091262	56	0.456311	0.048100	45	0.563107	0.008460
9	9	8	68	0.392857	0.097772	62	0.446429	0.047751	51	0.544643	0.009469
9	9	9	74	0.388430	0.093398	68	0.438017	0.047321	56	0.537190	0.008904
10	1	1	0	1.000000	0.045455	0	1.000000	0.045455			
10	2	1	3	0.812500	0.072261	2	0.875000	0.039627	0	1.000000	0.006993
10	2	2	7	0.681818	0.095238	5	0.772727	0.041292	2	0.909091	0.007326
10	3	1	6	0.714286	0.074925	5	0.761905	0.049950	2	0.904762	0.009491
10	3	2	10	0.642857	0.079853	8	0.714286	0.039361	4	0.857143	0.006061
10	3	3	14	0.588235	0.082105	12	0.647059	0.044843	7	0.794118	0.006581
10	4	1	10	0.629630	0.094439	8	0.703704	0.049817	4	0.851852	0.009058
10	4	2	14	0.588235	0.087796	12	0.647059	0.049534	7	0.794118	0.007709
10	4	3	18	0.560976	0.080364	16	0.609756	0.047764	11	0.731707	0.009629
10	4	4	23	0.520833	0.090451	20	0.583333	0.045339	14	0.708333	0.007948
10	5	1	13	0.593750	0.085331	11	0.656250	0.048701	6	0.812500	0.007992
10	5	2	18	0.550000	0.092437	15	0.625000	0.043398	10	0.750000	0.008731
10	5	3	23	0.510638	0.096662	20	0.574468	0.049272	14	0.702128	0.009033
10	5	4	28	0.490909	0.097473	24	0.563636	0.042664	18	0.672727	0.009027
10	5	5	33	0.467742	0.096851	29	0.532258	0.045909	22	0.645161	0.008935
10	6	1	17	0.552632	0.096918	14	0.631579	0.046615	9	0.763158	0.009887
10	6	2	22	0.521739	0.095008	19	0.586957	0.048928	13	0.717391	0.009192
10	6	3	27	0.500000	0.091298	24	0.555556	0.049602	17	0.685185	0.008356
10	6	4	32	0.483871	0.086315	29	0.532258	0.049122	22	0.645161	0.009852
10	6	5	38	0.457143	0.095262	34	0.514286	0.048127	26	0.628571	0.008761
10	6	6	43	0.448718	0.088541	39	0.500000	0.046806	31	0.602564	0.009842
10	7	1	20	0.534884	0.087281	17	0.604651	0.044752	11	0.744186	0.008261
10	7	2	26	0.500000	0.096557	22	0.576923	0.042974	16	0.692308	0.009396
10	7	3	31	0.483333	0.086789	28	0.533333	0.049625	20	0.666667	0.007738
10	7	4	37	0.463768	0.090962	33	0.521739	0.045577	25	0.637681	0.008213

Continued on next page

Table 9: Critical values and exact p-values of the Concordance coefficient τ_c for $k=3$ samples.

Table 9 – continued from previous page

Sample Sizes	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value
10 7 5	43	0.441558	0.093529	39	0.493506	0.049803	30	0.610390	0.008552
10 7 6	49	0.430233	0.094802	44	0.488372	0.045397	35	0.593023	0.008781
10 7 7	55	0.414894	0.095476	50	0.468085	0.048288	40	0.574468	0.008936
10 8 1	24	0.510204	0.095314	20	0.591837	0.042629	14	0.714286	0.009362
10 8 2	30	0.482759	0.097293	26	0.551724	0.046727	19	0.672414	0.009415
10 8 3	36	0.462687	0.096898	32	0.522388	0.049214	24	0.641791	0.009154
10 8 4	42	0.447368	0.094647	37	0.513158	0.042505	29	0.618421	0.008738
10 8 5	48	0.435294	0.091545	43	0.494118	0.043502	34	0.600000	0.008310
10 8 6	54	0.425532	0.088149	49	0.478723	0.043991	40	0.574468	0.009585
10 8 7	61	0.407767	0.095600	55	0.466019	0.044145	45	0.563107	0.009021
10 8 8	67	0.401786	0.091405	61	0.455357	0.044080	50	0.553571	0.008511
10 9 1	27	0.500000	0.086557	24	0.555556	0.049882	16	0.703704	0.007882
10 9 2	34	0.468750	0.097666	30	0.531250	0.049878	22	0.656250	0.009353
10 9 3	40	0.452055	0.091779	36	0.506849	0.048772	27	0.630137	0.008369
10 9 4	47	0.433735	0.097637	42	0.493976	0.046800	33	0.602410	0.009150
10 9 5	53	0.423913	0.089682	48	0.478261	0.044758	39	0.576087	0.009781
10 9 6	60	0.411765	0.092986	55	0.460784	0.048988	44	0.568627	0.008606
10 9 7	67	0.396396	0.095498	61	0.450450	0.046335	50	0.549550	0.009061
10 9 8	74	0.388430	0.097309	68	0.438017	0.049566	56	0.537190	0.009435
10 9 9	81	0.376923	0.098701	74	0.430769	0.046792	62	0.523077	0.009745
10 10 1	31	0.483333	0.092777	27	0.550000	0.047070	19	0.683333	0.008615
10 10 2	38	0.457143	0.097689	33	0.528571	0.044350	25	0.642857	0.009229
10 10 3	45	0.437500	0.099986	40	0.500000	0.048153	31	0.612500	0.009472
10 10 4	51	0.433333	0.088023	46	0.488889	0.043698	37	0.588889	0.009472
10 10 5	59	0.410000	0.098994	53	0.470000	0.045715	43	0.570000	0.009375
10 10 6	66	0.400000	0.097217	60	0.454545	0.047170	49	0.554545	0.009231
10 10 7	73	0.391667	0.095144	67	0.441667	0.048198	55	0.541667	0.009064
10 10 8	80	0.384615	0.092953	74	0.430769	0.048911	61	0.530769	0.008881
10 10 9	88	0.371429	0.099700	81	0.421429	0.049390	68	0.514286	0.009996
10 10 10	95	0.366667	0.096934	88	0.413333	0.049695	74	0.506667	0.009709

Table 9: Critical values and exact p-values of the Concordance coefficient τ_c for $k=3$ samples.

Sample Sizes	.10			.05			.01		
	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value
2 2 1 1									
2 2 2 1	0	1.000000	0.038095	0	1.000000	0.038095			
2 2 2 2	2	0.833333	0.095238	1	0.916667	0.038095	0	1.000000	0.009524
3 1 1 1									
3 2 1 1	0	1.000000	0.057143						
3 2 2 1	1	0.909091	0.050000	0	1.000000	0.014286			
3 2 2 2	3	0.800000	0.077778	2	0.866667	0.036508	0	1.000000	0.003175
3 3 1 1	1	0.888889	0.075000	0	1.000000	0.021429			
3 3 2 1	3	0.785714	0.097619	2	0.857143	0.046429	0	1.000000	0.004762
3 3 2 2	5	0.722222	0.097143	3	0.833333	0.027619	1	0.944444	0.003810
3 3 3 1	4	0.750000	0.069286	3	0.812500	0.034286	1	0.937500	0.005714
3 3 3 2	7	0.681818	0.096883	5	0.772727	0.034805	3	0.863636	0.008442
3 3 3 3	9	0.640000	0.084091	7	0.720000	0.034221	4	0.840000	0.005325
4 1 1 1									
4 2 1 1	1	0.900000	0.085714	0	1.000000	0.028571			
4 2 2 1	2	0.857143	0.055556	1	0.928571	0.022222	0	1.000000	0.006349
4 2 2 2	4	0.777778	0.064444	3	0.833333	0.033016	1	0.944444	0.005079
4 3 1 1	2	0.833333	0.076190	1	0.916667	0.033333	0	1.000000	0.009524
4 3 2 1	4	0.764706	0.077619	3	0.823529	0.041429	1	0.941176	0.007143
4 3 2 2	6	0.727273	0.068312	5	0.772727	0.040173	2	0.909091	0.004329
4 3 3 1	6	0.714286	0.081299	5	0.761905	0.048571	2	0.904762	0.005584
4 3 3 2	9	0.653846	0.092309	7	0.730769	0.038615	4	0.846154	0.006342
4 3 3 3	11	0.645161	0.071618	10	0.677419	0.048871	6	0.806452	0.006678
4 4 1 1	3	0.812500	0.060952	2	0.875000	0.032381	0	1.000000	0.003810
4 4 2 1	6	0.714286	0.089004	4	0.809524	0.031169	2	0.904762	0.007100
4 4 2 2	8	0.692308	0.068283	7	0.730769	0.043579	4	0.846154	0.007734
4 4 3 1	8	0.680000	0.078672	6	0.760000	0.030996	4	0.840000	0.009264
4 4 3 2	11	0.645161	0.078326	9	0.709677	0.035791	6	0.806452	0.007792
4 4 3 3	14	0.611111	0.077325	12	0.666667	0.038965	8	0.777778	0.006591

Continued on next page

Table 10: Critical values and exact p-values of the Concordance coefficient τ_c for $k=4$ samples.

Table 10 – continued from previous page

Sample Sizes	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value	<i>dis</i>	τ_c	p-value
4 4 4 1	11	0.633333	0.095984	9	0.700000	0.045594	5	0.833333	0.005808
4 4 4 2	14	0.611111	0.084038	12	0.666667	0.043035	8	0.777778	0.007570
4 4 4 3	18	0.571429	0.097366	15	0.642857	0.040186	11	0.738095	0.008775
4 4 4 4	21	0.562500	0.083959	19	0.604167	0.049523	14	0.708333	0.009514
5 1 1 1	0	1.000000	0.071429						
5 2 1 1	1	0.909091	0.047619	1	0.909091	0.047619			
5 2 2 1	3	0.812500	0.058730	2	0.875000	0.028571	0	1.000000	0.003175
5 2 2 2	6	0.714286	0.091631	4	0.809524	0.030592	2	0.904762	0.006638
5 3 1 1	3	0.785714	0.076190	2	0.857143	0.040476	0	1.000000	0.004762
5 3 2 1	5	0.750000	0.065584	4	0.800000	0.037662	2	0.900000	0.008874
5 3 2 2	8	0.680000	0.079221	6	0.760000	0.031025	4	0.840000	0.009235
5 3 3 1	8	0.652174	0.092388	6	0.739130	0.036905	3	0.869565	0.005519
5 3 3 2	11	0.633333	0.089419	9	0.700000	0.041492	6	0.800000	0.009232
5 3 3 3	14	0.588235	0.087484	12	0.647059	0.044634	8	0.764706	0.007746
5 4 1 1	5	0.722222	0.087446	3	0.833333	0.030303	1	0.944444	0.006061
5 4 2 1	8	0.666667	0.099279	6	0.750000	0.041631	3	0.875000	0.006854
5 4 2 2	11	0.633333	0.096947	9	0.700000	0.045865	5	0.833333	0.005972
5 4 3 1	10	0.655172	0.077312	8	0.724138	0.034466	5	0.827586	0.007126
5 4 3 2	14	0.600000	0.093723	12	0.657143	0.048620	8	0.771429	0.008841
5 4 3 3	17	0.585366	0.082249	15	0.634146	0.045133	10	0.756098	0.006435
5 4 4 1	13	0.617647	0.082489	11	0.676471	0.041660	7	0.794118	0.006974
5 4 4 2	17	0.585366	0.088025	15	0.634146	0.048942	10	0.756098	0.007269
5 4 4 3	21	0.553191	0.091439	18	0.617021	0.040930	13	0.723404	0.007254
5 4 4 4	25	0.537037	0.091748	22	0.592593	0.044712	17	0.685185	0.009977
5 5 1 1	7	0.666667	0.098966	5	0.761905	0.041126	2	0.904762	0.006854
5 5 2 1	10	0.642857	0.094933	8	0.714286	0.044483	4	0.857143	0.005606
5 5 2 2	13	0.617647	0.083004	11	0.676471	0.041903	7	0.794118	0.007191
5 5 3 1	13	0.593750	0.093169	11	0.656250	0.047627	7	0.781250	0.008432
5 5 3 2	17	0.575000	0.097342	14	0.650000	0.039675	10	0.750000	0.008422
5 5 3 3	20	0.555556	0.078592	18	0.600000	0.045644	13	0.711111	0.008304
5 5 4 1	16	0.589744	0.086815	14	0.641026	0.047844	9	0.769231	0.006849
5 5 4 2	20	0.565217	0.083163	18	0.608696	0.048813	13	0.717391	0.009174
5 5 4 3	25	0.528302	0.099365	22	0.584906	0.048982	16	0.698113	0.007882
5 5 4 4	29	0.516667	0.091307	26	0.566667	0.047580	20	0.666667	0.009275
5 5 5 1	19	0.558140	0.082239	17	0.604651	0.047987	12	0.720930	0.008799
5 5 5 2	24	0.538462	0.091195	21	0.596154	0.044209	16	0.692308	0.009718
5 5 5 3	29	0.500000	0.098583	25	0.568966	0.040931	19	0.672414	0.007511
5 5 5 4	33	0.507463	0.084597	30	0.552239	0.046171	23	0.656716	0.007853
5 5 5 5	38	0.479452	0.088106	34	0.534247	0.041674	27	0.630137	0.008096
6 1 1 1	0	1.000000	0.047619	0	1.000000	0.047619			
6 2 1 1	2	0.857143	0.066667	1	0.928571	0.028571	0	1.000000	0.009524
6 2 2 1	4	0.789474	0.060462	3	0.842105	0.032468	1	0.947368	0.006061
6 2 2 2	7	0.708333	0.077201	6	0.750000	0.048485	3	0.875000	0.007504
6 3 1 1	4	0.764706	0.075325	3	0.823529	0.042857	1	0.941176	0.009091
6 3 2 1	7	0.695652	0.088095	5	0.782609	0.033983	3	0.869565	0.009848
6 3 2 2	10	0.655172	0.087568	8	0.724138	0.039494	5	0.827586	0.008225
6 3 3 1	10	0.642857	0.099459	8	0.714286	0.045538	4	0.857143	0.005370
6 3 3 2	13	0.617647	0.085707	11	0.676471	0.043064	7	0.794118	0.007064
6 3 3 3	16	0.600000	0.075737	14	0.650000	0.040621	10	0.750000	0.008509
6 4 1 1	6	0.727273	0.073737	5	0.772727	0.046609	2	0.909091	0.007792
6 4 2 1	9	0.678571	0.074026	7	0.750000	0.032534	4	0.857143	0.006460
6 4 2 2	13	0.617647	0.092254	11	0.676471	0.047099	7	0.794118	0.008249
6 4 3 1	12	0.636364	0.074599	10	0.696970	0.036371	7	0.787879	0.009576
6 4 3 2	16	0.600000	0.080891	14	0.650000	0.043967	10	0.750000	0.009549
6 4 3 3	20	0.565217	0.085039	18	0.608696	0.049821	13	0.717391	0.009305
6 4 4 1	16	0.589744	0.094578	13	0.666667	0.037925	9	0.769231	0.007777
6 4 4 2	20	0.565217	0.090522	17	0.630435	0.040149	12	0.739130	0.006927
6 4 4 3	24	0.547170	0.085709	21	0.603774	0.040995	16	0.698113	0.008781
6 4 4 4	29	0.516667	0.097941	25	0.583333	0.040644	19	0.683333	0.007460
6 5 1 1	8	0.680000	0.073704	7	0.720000	0.049728	3	0.880000	0.006494
6 5 2 1	12	0.625000	0.089767	10	0.687500	0.045692	6	0.812500	0.007992
6 5 2 2	16	0.589744	0.095524	13	0.666667	0.038420	9	0.769231	0.008052
6 5 3 1	15	0.605263	0.079615	13	0.657895	0.042893	9	0.763158	0.009229
6 5 3 2	20	0.555556	0.098821	17	0.622222	0.044475	12	0.733333	0.007945
6 5 3 3	24	0.538462	0.093325	21	0.596154	0.045191	16	0.692308	0.009911
6 5 4 1	19	0.568182	0.088703	16	0.636364	0.038969	12	0.727273	0.009856
6 5 4 2	24	0.538462	0.098090	21	0.596154	0.048090	15	0.711538	0.007604

Continued on next page

Table 10: Critical values and exact p-values of the Concordance coefficient τ_c for $k=4$ samples.

Table 10 – continued from previous page

Sample Sizes	dis	τ_c	p-value	dis	τ_c	p-value	dis	τ_c	p-value
6 5 4 3	28	0.525424	0.085898	25	0.576271	0.044131	19	0.677966	0.008293
6 5 4 4	33	0.507463	0.090101	30	0.552239	0.049583	23	0.656716	0.008625
6 5 5 1	23	0.540000	0.096573	20	0.600000	0.047042	14	0.720000	0.007284
6 5 5 2	28	0.517241	0.097004	24	0.586207	0.040080	18	0.689655	0.007270
6 5 5 3	33	0.500000	0.096429	29	0.560606	0.043157	23	0.651515	0.009514
6 5 5 4	38	0.486486	0.093108	34	0.540541	0.044474	27	0.635135	0.008812
6 5 5 5	43	0.475610	0.090015	39	0.524390	0.045390	31	0.621951	0.008195
6 6 1 1	11	0.633333	0.097617	8	0.733333	0.034775	5	0.833333	0.009134
6 6 2 1	14	0.621622	0.076902	12	0.675676	0.041463	8	0.783784	0.008910
6 6 2 2	19	0.568182	0.097046	16	0.636364	0.043344	11	0.750000	0.007713
6 6 3 1	18	0.581395	0.081966	16	0.627907	0.047471	11	0.744186	0.008665
6 6 3 2	23	0.549020	0.091718	20	0.607843	0.044161	15	0.705882	0.009600
6 6 3 3	28	0.517241	0.099031	24	0.586207	0.040890	18	0.689655	0.007417
6 6 4 1	22	0.560000	0.082630	19	0.620000	0.038997	14	0.720000	0.008129
6 6 4 2	27	0.534483	0.084812	24	0.586207	0.043401	18	0.689655	0.008070
6 6 4 3	32	0.515152	0.085020	29	0.560606	0.046268	22	0.666667	0.007791
6 6 4 4	38	0.486486	0.098717	34	0.540541	0.047595	27	0.635135	0.009613
6 6 5 1	26	0.535714	0.082947	23	0.589286	0.042216	17	0.696429	0.007727
6 6 5 2	32	0.507692	0.095002	28	0.569231	0.042368	22	0.661538	0.009272
6 6 5 3	37	0.493151	0.088417	33	0.547945	0.041732	26	0.643836	0.008054
6 6 5 4	43	0.475610	0.094851	39	0.524390	0.048227	31	0.621951	0.008874
6 6 5 5	48	0.466667	0.086446	44	0.511111	0.045656	36	0.600000	0.009533
6 6 6 1	31	0.507936	0.098636	27	0.571429	0.044191	21	0.666667	0.009750
6 6 6 2	36	0.500000	0.087213	32	0.555556	0.041053	25	0.652778	0.007871
6 6 6 3	42	0.481481	0.090321	38	0.530864	0.045479	30	0.629630	0.008168
6 6 6 4	48	0.466667	0.090941	44	0.511111	0.048376	35	0.611111	0.008207
6 6 6 5	54	0.454545	0.090669	49	0.505051	0.043100	40	0.595960	0.008145
6 6 6 6	60	0.444444	0.089781	55	0.490741	0.044896	46	0.574074	0.009741

Table 10: Critical values and exact p-values of the Concordance coefficient τ_c for $k=4$ samples.

Javier Alcaraz
Center of Operations Research, Miguel Hernández University
Alda. Universidad sn, 03202 Elche (Spain)
E-mail: jalcaraz@umh.es

Laura Anton-Sanchez
Center of Operations Research, Miguel Hernández University
Alda. Universidad sn, 03202 Elche (Spain)
E-mail: l.anton@umh.es

Juan Francisco Monge
Center of Operations Research, Miguel Hernández University
Alda. Universidad sn, 03202 Elche (Spain)
E-mail: monge@umh.es

htestClust: A Package for Marginal Inference of Clustered Data Under Informative Cluster Size

by Mary Gregg, Somnath Datta and Douglas Lorenz

Abstract When observations are collected in/organized into observational units, within which observations may be dependent, those observational units are often referred to as "clustered" and the data as "clustered data". Examples of clustered data include repeated measures or hierarchical shared association (e.g., individuals within families). This paper provides an overview of the R package **htestClust**, a tool for the marginal analysis of such clustered data with potentially informative cluster and/or group sizes. Contained in **htestClust** are clustered data analogues to the following classical hypothesis tests: rank-sum, signed rank, t -, one-way ANOVA, F, Levene, Pearson/Spearman/Kendall correlation, proportion, goodness-of-fit, independence, and McNemar. Additional functions allow users to visualize and test for informative cluster size. This package has an easy-to-use interface mimicking that of classical hypothesis-testing functions in the R environment. Various features of this package are illustrated through simple examples.

1 Introduction

Observations often occur or can be organized into units called clusters, within which those observations may be dependent. For example, individuals may be repeatedly assessed or naturally belong to some hierarchical structure like a family unit. Potential correlation among intra-cluster observations clearly invalidates the use of classical hypothesis tests for the analysis of such data. Instead, inference is generally performed using model-based methods that capture intra-cluster relationships through parametric or semi-parametric assumptions. Generalized estimating equations (GEEs) are one such approach that fit marginal generalized linear models to clustered data while making a working assumption on the correlation structure. GEE models are appealing for their flexible and robust nature, and several packages in the R environment, such as **gee** (Carey et al., 2019) and **geepack** (Halekoh et al., 2006), offer an implementation of this method. However, GEEs and other standard methods for analysis of clustered data operate under an assumption that the number of observations within the clusters (defined as the cluster size) is ignorable. In practice, this assumption may not hold and cluster size may vary systematically in a way that carries information related to the response of interest. When this occurs data are said to have informative cluster size (ICS). Examples of ICS can be found in data related to dental health (Williamson et al., 2003), pregnancy studies (Chaurasia et al., 2018), and longitudinal rehabilitation (Lorenz et al., 2011), among others. For data with ICS, standard model-based methods can produce biased inference as their estimates may be overweighted in favor of larger clusters.

A related but distinct type of informativeness occurs when the distribution of group-defining covariates varies in a way that carries information on the response. Such phenomenon has been called informative within-cluster group size (IWCGS), as well as informative covariate structure (Pavlou, 2012), sub-cluster covariate informativeness (Lorenz et al., 2018), and informative intra-cluster group size (Dutta and Datta, 2016a). This additional informativeness may occur simultaneously with or separately from ICS, and similarly can result in the failure of standard methods to maintain appropriate nominal size (Huang and Leroux, 2011; Dutta and Datta, 2016a).

Williamson et al. (2003) developed a reweighting methodology that corrects for potential bias from cluster- or group-size informativeness. This reweighting originates from a Monte Carlo resampling process, and leads to weighting observations proportional to their inverse cluster or within-cluster group size. Correction for ICS/IWCGS was originally proposed in the context of modeling, and a number of extensions to this application have been established (Bible et al., 2016; Iosif and Sampson, 2014; Mitani et al., 2019, 2020). However, when adjustment for covariates is not of interest, this reweighting can be directly applied in the estimation of marginal parameters. Under mild conditions, such estimates are asymptotically normal, permitting Wald-type intervals and tests. This methodology has been applied to develop rank-based tests (Datta and Satten, 2005, 2008; Dutta and Datta, 2016a), and tests of correlation (Lorenz et al., 2011), proportions (Gregg et al., 2020), means and variances (Gregg, 2020). This collection of reweighted non-model-based hypothesis tests includes clustered data analogues of the following classical tests: rank-sum, signed rank, t -, one-way ANOVA, F, Levene, Pearson/Spearman/Kendall correlation, proportion, goodness-of-fit, independence, and McNemar.

These clustered data analogues to standard hypothesis tests provide simple and intuitive means of

performing exploratory and preliminary analysis of clustered data in which the cluster and/or group size varies and is potentially informative. However, many of these tests are recent developments that are not available in a software environment. We address this deficiency through the package **hstestClust**, the first R package designed as a comprehensive collection of direct, non-model-based inferential methods for analysis of clustered data with potential ICS and/or IWCGS. Introduced in this paper, **hstestClust** implements the collection of methods by [Datta and Satten \(2005, 2008\)](#); [Dutta and Datta \(2016a\)](#); [Lorenz et al. \(2011\)](#); [Gregg et al. \(2020\)](#) and [Gregg \(2020\)](#), as well as a method by [Nevalainen et al. \(2017\)](#) that tests for the presence of informative cluster size. The syntax and output of functions contained in **hstestClust** are intentionally modeled after their corresponding analogous classical function, allowing researchers to assess various marginal analyses through intuitive and user-friendly means. The rest of this paper is organized as follows. We will begin by briefly summarizing the reweighting approach developed by [Williamson et al. \(2003\)](#) and describe how its application has been used in the development of hypothesis tests of marginal parameters in clustered data. We will then provide an overview of the **hstestClust** package, describe the features and structure of functions, and describe an illustrative simulated data set with informativeness. Finally, we will demonstrate **hstestClust** using the example data set and close with a discussion.

2 Methods for clustered data under informativeness

In this section we outline the weighting methodology that corrects for bias from ICS and IWCGS, and describe the general form of the tests in **hstestClust** that implement this weighting. We then summarize the balanced bootstrap design implemented in the test of ICS by [Nevalainen et al. \(2017\)](#).

Notation

Consider a sample of M independent clusters, with each cluster containing n_i potentially correlated observations, $i = 1, \dots, M$. The j^{th} observation from cluster i is X_{ij} , with $j = 1, \dots, n_i$. The collection of data from cluster i is $V_i = \{n_i, X_{i1}, \dots, X_{in_i}\}$ and the set of all observed data is $V = \{V_1, \dots, V_M\}$. Informative cluster size is defined as inequality between the marginal distribution of the response X and the distribution of X conditional on cluster size: $P(X_{ij} \leq x \mid n_i = n) \neq P(X_{ij} \leq x), n = 1, 2, \dots; j = 1, \dots, n_i$.

When observations within clusters belong to one of K distinct groups, we define the variable $G_{ij} = k$ to represent that observation j from cluster i belongs to group k , $k = 1, \dots, K$. We let $n_i^{(k)}$ denote the number of observations from cluster i in group k , and note that $n_i = \sum_{k=1}^K n_i^{(k)}$. We define $K_i^c = \sum_{k=1}^K I[n_i^{(k)} > 0]$ to be the number of distinct groups observed in cluster i . When $K_i^c < K$, not all groups are observed in cluster i , a condition referred to as incomplete group structure. The data from cluster i is now the set $V_i = \{n_i^{(k)}, (X_{ij}, G_{ij})\}$, with observations belonging to group k denoted as the set $\{X_{i1}^{(k)}, \dots, X_{in_i^{(k)}}^{(k)}\}$. Informative within-cluster group size can be defined as $P(X_{ij} \leq x \mid n_i^{(k)}) \neq P(X_{ij} \leq x)$, i.e. that the marginal distribution of X differs from the distribution of X conditional on the within-cluster group size.

Weighting for ICS/IWCGS

Let θ denote a marginal parameter to be estimated and/or tested. One approach for estimating θ is within-cluster resampling (WCR), in which one observation is randomly selected from each cluster ([Hoffman et al., 2001](#)). The resulting subset of data, $\mathbf{X}^* = \{X_1^*, X_2^*, \dots, X_M^*\}$, consists of independent observations so an estimate of the parameter, $\hat{\theta}$, can be calculated using standard i.i.d. methods. Clearly, this estimate is inefficient, using only a subset of the data, so the resampling process is repeated many times, creating many pseudo data sets and estimates $\hat{\theta}_q^*$. An overall estimate of θ is obtained over Q resamplings (Q large) by averaging the resampled estimates, $\hat{\theta}^* = \frac{1}{Q} \sum_{q=1}^Q \hat{\theta}_q^*$. This estimator was shown to be asymptotically normal and inference can be conducted using Wald-type intervals and tests.

The method of reweighting proposed by [Williamson et al. \(2003\)](#) derives from WCR by noting that as $M, Q \rightarrow \infty$, the overall resampled estimator converges to $\hat{\theta} = E[\hat{\theta}_q^* \mid V]$ with respect to the resampling distribution. This marginalization is equivalent to averaging the resampled estimator across all realizations of the resampled data. As sampling is uniform across clusters and with equal

probability within each cluster, each observation is weighted by the inverse of the associated cluster size.

The link between WCR and reweighting can be illustrated by a simple example - estimating a marginal mean. For a single resampled data set produced by WCR, the estimate of the marginal mean is the simple average, $\hat{\theta}_q^* = \frac{1}{M} \sum_{i=1}^M X_i^*$. Application of the marginalization calculation produces

$$\begin{aligned} \hat{\theta} &= E \left[\hat{\theta}_q^* \mid \mathbf{V} \right] \\ &= \frac{1}{M} \sum_{i=1}^M E \left[X_i^* \mid \mathbf{V} \right] = \frac{1}{M} \sum_{i=1}^M \frac{1}{n_i} \sum_{j=1}^{n_i} X_{ij} \end{aligned}$$

The independence of clusters allows the expectation of the resampled estimate to be expressed as the average of the expectations. Conditioned on the observed data \mathbf{V} , the expectation of a resampled observation from a particular cluster is the average of all observations from the cluster, as the WCR process resamples observations from that cluster with equal probability.

The weighting that corrects for ICS can be adapted to correct for IWCGS by modifying the underlying resampling process into a two-step procedure that marginalizes the within-cluster distribution of groups (Dutta and Datta, 2016a; Huang and Leroux, 2011). In this two-step resampling, we first select a group, G_i^* , with uniform probability from the levels of G available in cluster i . Second, we select X_i^* from the set of observations in group k , $\{X_{i1}^{(k)}, \dots, X_{in_i^{(k)}}^{(k)}\}$, where k is the group selected in the first step of the process. As in the original WCR methodology, this process is repeated for all clusters, resulting in a resampled data $(\mathbf{X}^*, \mathbf{G}^*) = \{(X_1^*, G_1^*), \dots, (X_M^*, G_M^*)\}$. An estimate of the parameter of interest is calculated from this resampled data. When the marginalization calculation is applied to a single WCR estimate produced by this two-step process, observations are weighted by the product of the two selection probabilities - one for the selection of a group and one for the selection of an observation within the group. Since both of these selections are made with equal probability, the weights in a given cluster are defined by the number of groups available in that cluster and the number of observations within the group:

$$w_{ij} = \begin{cases} \left(K_i^c n_i^{(k)} \right)^{-1}, & \text{if } n_i^{(k)} > 0 \\ 0, & \text{otherwise.} \end{cases}$$

Hypothesis tests of marginal parameters

The asymptotic normality of the estimators described in the previous section has been established under mild regularity conditions (Datta and Satten, 2005, 2008; Williamson et al., 2003). The tests of ranks, correlation, proportions, means and variances contained in **htestClust** all leverage this asymptotic normality through the general univariate and multivariate Wald-type forms

$$Z = \frac{S - E[S]}{\sqrt{\hat{V}(S)}} \quad \mathbf{X} = (\mathbf{S} - E(\mathbf{S}))^T (\hat{\mathbf{V}}(\mathbf{S}))^{-1} (\mathbf{S} - E(\mathbf{S})).$$

The statistic, S , differs across the various tests. However, in each of the tests S is either a reweighted estimator derived through the marginalization calculation or a smooth function of such reweighted estimators. $E[S]$ is the statistic's expected value under the null hypothesis and $\hat{V}(S)$ is an estimate of the variance of S . Z asymptotically follows a standard normal distribution, while \mathbf{X} asymptotically follows a chi square distribution with $K - 1$ degrees of freedom.

Methods of estimating the variance of S also vary across the tests. The rank-sum and signed rank tests weighted for ICS apply Hajek projections (Datta and Satten, 2005, 2008), while the tests of correlation use an approach based on the empirical variances of within-cluster averages (Lorenz et al., 2011). The rank-sum test weighted for IWCGS and the multi-group tests of means and variances use jackknife estimates (Dutta and Datta, 2016a; Gregg, 2020). The tests of proportions were constructed and evaluated under different variance estimation techniques including sandwich forms, method of moments, and empirical estimates. Gregg et al. (2020) provide a detailed examination by simulation of different variance estimation techniques in the context of estimating and testing proportions, and note that no one variance estimation technique is optimal for different types of tests. Further, the size and power of the tests in **htestClust** previously have been evaluated via simulation in the source manuscripts for each test. Predictably, each has been shown to perform well under the informativeness conditions for which they were designed to adjust.

Testing for informative cluster size

Nevalainen et al. (2017) proposed a test for ICS using a novel balanced bootstrap scheme. As it might be desirable to perform this test prior to the application of the marginal methods mentioned thus far, we have included this test for ICS in the **htestClust** package and briefly summarize it below.

Let $V = (V_1, \dots, V_M)$ be a collection of independent clustered observations, where $V_i = (n_i; X_{i1}, \dots, X_{in_i})$ is the data from cluster i . Assuming exchangeability of observations within clusters, the hypothesis of interest is $H_0 : P(X_{ij} \leq x | n_i = k) = F(x), k = 1, 2, \dots; j = 1, \dots, k$, for some unknown distribution F . Two test statistics are proposed for testing H_0 ; a Kolmogorov-Smirnov type statistic takes the form

$$T_F = \sup_x |\hat{F}(x) - \tilde{F}(x)|$$

where $\hat{F}(x) = \frac{1}{n} \sum_{i=1}^M \sum_{j=1}^{n_i} I[X_{ij} \leq x]$ and $\tilde{F}(x) = \frac{1}{M} \sum_{i=1}^M \frac{1}{n_i} \sum_{j=1}^{n_i} I[X_{ij} \leq x]$. A Cramer-von Mises type alternative to T_F is:

$$T_{CM} = \sum_{k \in \psi} \left[k M_k \int (\hat{F}_k(x) - \hat{F}(x))^2 dx \right],$$

where ψ represents the set of unique cluster sizes, M_k represents the number of clusters of size k , and $\hat{F}_k(x) = \frac{1}{k M_k} \sum_{i=1}^M \sum_{j=1}^{n_i} I[n_i = k, X_{ij} \leq x]$. T_{CM} is suggested for use when there is a small number of distinct cluster sizes, as it tends to be more powerful. T_F is preferred when the number of distinct cluster sizes is large and the number of clusters with those sizes is small, as T_{CM} tends to be too liberal.

The bootstrap scheme, which is employed for either statistic, is as follows. For iteration $b, b = 1, \dots, B$,

1. Permute observations within each cluster.
2. Resample clusters from the permuted data by performing the following for $i = 1, \dots, M$:
 - (a) Randomly select a cluster $i^*, i^* = 1, \dots, M$.
 - (b) If $n_{i^*} \geq n_i$, form the i^{th} bootstrapped cluster from the first n_i observation from cluster i^* ; e.g., $V_{bi}^* = (n_i; X_{i^*1}, \dots, X_{i^*n_i})$.
 - (c) If $n_{i^*} < n_i$, form the i^{th} bootstrapped cluster by merging observations from the resampled cluster i^* and observations from the closest 'matching' cluster to cluster i^* ; e.g., $V_{bi}^* = (n_i; X_{i^*1}, \dots, X_{i^*n_{i^*}}, X_{k(n_{i^*}+1)}, \dots, X_{kn_i})$, where $k = \arg \min_k \{D(V_{i^*}, V_k) : n_k \geq n_i\}$. The closest matching cluster is determined by minimum distance calculated by $D(V_i, V_j) = (\min\{n_i, n_j\})^{-1} \sum_{k=1}^{\min\{n_i, n_j\}} (X_{ik} - X_{jk})^2$.
3. Calculate the test statistic from the collection of bootstrapped clusters, $T_b^* = T(V_b^*), V_b^* = (V_{b1}^*, \dots, V_{bM}^*)$.

The approximate p-value is then obtained from the sample of bootstrapped test statistics by $\frac{1}{B} \sum_{b=1}^B I[T_b^* \geq T]$, where T is the desired test statistic calculated from the original data.

3 Overview of htestClust

htestClust includes ten functions for conducting different hypothesis tests under ICS, one function for visualizing informativeness in cluster size, and a simulated hypothetical data set to illustrate the use of the functions. We first note that, at the time of this publication, we are aware of only two other R packages available on CRAN that provide functions for analyzing data under ICS and IWCGS: **clusrank** (Jiang, 2018) and **ClusterRankTest** (Dutta and Datta, 2016b). Each of these packages provides functionality only for rank-based tests for clustered data, i.e. clustered data analogues of the well-known Wilcoxon signed rank and rank sum tests. We know of no other R package that includes the broad range of tests of means, proportions, variances, and correlations in addition to these rank-based tests that is provided by **htestClust**.

Package functions, syntax, and output

With the exception of the test of informative cluster size, each of the hypothesis testing functions implemented in **htestClust** has a well-known analogue test for i.i.d. data (Table 1). As such, the syntax and output of the functions in **htestClust** are designed to conform with that of the analogous i.i.d. functions from the R **stats** library. A notable but necessary departure from this correspondence is that

hstestClust function	Reweighted test(s)	Classical analogue function
<code>chisqtestClust()</code>	Chi squared goodness of fit, independence	<code>chisq.test()</code>
<code>cortestClust()</code>	Correlation	<code>cor.test()</code>
<code>icstestClust()</code>	Test of ICS	NA
<code>levenetestClust()</code>	K-group test of variance	<code>leveneTest()</code>
<code>mcnemartestClust()</code>	Homogeneity	<code>mcnemar.test()</code>
<code>onewaytestClust()</code>	K-group mean equality	<code>oneway.test()</code>
<code>proptestClust()</code>	Proportion	<code>prop.test()</code>
<code>ttestClust()</code>	Test of means (one/two group, paired)	<code>t.test()</code>
<code>var.testClust()</code>	2-group test of variance	<code>var.test()</code>
<code>wilcoxtestClust()</code>	Rank sum, signed rank	<code>wilcox.test()</code>

Table 1: Hypothesis testing functions available in the **hstestClust** package. Each row gives the name of a **hstestClust** function, the reweighted test the function performs, and the R function that executes the corresponding classical analogue test. All classical analogue functions are available in R through the **stats** package, except for `leveneTest()`, which is included in the **car** package.

the **hstestClust** functions require as input (1) a variable identifying the clusters as an argument in the data set or (2) a cluster-level summary of the data.

As an example, consider the syntax for the **stats** and **hstestClust** functions for conducting a test of a single proportion:

```
prop.test(x, n, p = NULL, alternative = c("two.sided", "less", "greater"),
conf.level = 0.95, correct = TRUE)

proptestClust(x, id, p = NULL, alternative = c("two.sided", "less",
"greater"), variance = c("sand.null", "sand.est", "emp", "MoM"),
conf.level = 0.95)
```

The **stats** library function `prop.test` does not operate on variables in a data frame, but instead takes summary counts as its input. Argument `x` can be a scalar representing the number of binomial successes, whence `n` is required as the number of binomial trials. Alternatively, `x` can be a one-dimensional table or matrix with two entries, whence `n` is omitted. The remaining arguments customize the test in ways familiar to most users.

The function `proptestClust` from **hstestClust** operates on binary variables in a data frame or on cluster-level summary counts. In this function, `x` may be a binary variable measured over clusters, wherein `id` is required as a vector of cluster identifiers. Alternatively, `x` may instead be a two-dimensional table of within-cluster counts of failures and successes, wherein `id` is omitted. As previously noted, several options are available for variance estimation; these may be selected by the user through the `variance` argument. Additional customization of the test is as in `prop.test`.

Each of the testing functions in **hstestClust** has been constructed in this vein – parallel to the analogous **stats** function with contingencies necessary for clustered data. **hstestClust** functions accept vector input that designates the response, grouping (if necessary), and clustering variables. However, for convenience, many functions are designed with a secondary interface accepting tables or formulas. Like their **stats** package analogues, **hstestClust** testing functions produce `list` objects of class `hstest` for which the `print` method behaves in the usual way.

`icsPlot` provides a simple method for illustrating informative cluster size, providing a visual supplement to the results of the test of ICS, `icstestClust`. Briefly, `icsPlot` plots a within-cluster summary statistic of a variable, such as a mean, against the size of each cluster. For quantitative variables, `icsPlot` produces a scatterplot of a within-cluster measure of location (mean, median) or variation (SD, variance, IQR, range) against cluster size. For a categorical variable, a barplot of within-cluster proportions is produced.

Simulated example data set

hstestClust includes a simulated data set named `screen8` of clustered observations with informativeness, created under a hypothetical scenario we briefly describe here. A large school district has conducted a voluntary comprehensive exit survey for students graduating elementary school, collecting demographic, biometric, and academic performance data. The clustering mechanism for these data are the schools, with students comprising the observations within clusters.

The school district has offered an incentive program to boost participation, wherein schools having

Variable	Description
sch.id	School identification variable
stud.id	Student identification variable within school
age	Student age in years
gender	Student gender
height	Student height in inches
weight	Student weight in lbs
math	Student score on standardized math test
read	Student score on standardized reading test
phq2	Ordinal (0-6) score from a mental health screening. Higher scores correspond to higher levels of depression
qfit	Age-adjusted fitness quartile from physical health assessment taken at end of school year
qfit.s	Age-adjusted fitness quartile from physical health assessment taken at beginning of school year
activity	Student after-school activity

Table 2: Variables in screen8 data set. Each row gives the name of a variable included in the screen8 data set and its associated description.

higher participation rates are rewarded with priority status for classroom and technology upgrades for the new academic year. This incentive introduces the potential for ICS – resource-poor schools may exhibit greater participation (larger cluster sizes) but also tend to have students with poorer health metrics and standardized test scores.

screen8 contains data from 2224 students from 73 schools in this district. Cluster sizes – the number of students participating in the exit survey at each school – ranged from 17 to 50, with a median of 30. The first few lines of the data are printed below, followed by the tabulated number of participants from each school and a summary of the cluster sizes. Table 2 provides details on the variables in the data set.

```
R> library(htestClust)
R> data(screen8)
R> head(screen8)
sch.id stud.id age gender height weight math read phq2 qfit qfit.s activity
1      1      1  15  M     65   136  69  75   3  Q2    Q2   other
2      1      2  14  M     66   135  80  57   2  Q4    Q3   other
3      1      3  15  M     65   146  60  85   0  Q2    Q3   sports
4      1      4  15  M     68   156  70  83   1  Q3    Q2   other
5      1      5  15  M     68   170  66  60   1  Q2    Q2   sports
6      1      6  14  M     63   109  84  62   0  Q1    Q1  academic

R> (tab <- table(screen8$sch.id))
1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
35 32 26 33 23 25 27 21 39 28 32 38 35 24 29 27 36 29 38 39 25 30 36 29 46 27
...

R> summary(as.vector(tab))
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
17.00  25.00   30.00   30.47  36.00   50.00
```

4 Examples

In this section, we demonstrate usage of the functions in **htestClust** using the `screen8` data set. Our illustration is not comprehensive, but users can learn more about functions not covered here by browsing the associated help files. To motivate the demonstration, we'll investigate the following questions:

1. Is the proportion of students having "proficient" standardized math test scores (65 or greater) more than 0.75?
2. Are participation in extracurricular activity and gender independent?
3. Are mean standardized math test scores different between male and female students?
4. Are mean standardized reading test scores different among groups defined by extracurricular activities?

Evaluating informative cluster size

Before addressing these questions, we illustrate how to assess the potential informativeness of cluster size in the data set, starting by visualizing ICS through the `icsPlot` function. The arguments to `icsPlot` specify the variable of interest, a cluster-identifying variable, and a summary function to be applied to the variable within each cluster. This summary can be any of 'obs', 'mean', 'median', 'var', 'IQR', 'range', 'prop', producing plots of the observations themselves, measure of location, or measures of variation against cluster size. Option 'prop' can only be used when the variable of interest is a factor, so numerically coded categorical variables must be converted to factors. Standard R graphical parameters can also be specified when calling `icsPlot()`.

```
R> ### Figure 1
R> par(mfrow = c(1,2))
R> icsPlot(x = screen8$math, id = screen8$sch.id, FUN = "mean", pch = 20)
R> icsPlot(x = screen8$read, id = screen8$sch.id, FUN = "mean", pch = 20)

R> ### Figure 2
R> layout(mat = matrix(c(1, 2), nrow = 1, ncol = 2),
+         heights = c(1, 2), # Heights of the two rows
+         widths = c(2, 2.5))
R> par(mar = c(5, 4, 1, 0))
R> icsPlot(x = screen8$gender, id = screen8$sch.id, FUN = "prop",
+         ylab = "P(Female)", pch = 20)
R> par(mar = c(5, 4, 1, 5))
R> icsPlot(x = screen8$activity, id = screen8$sch.id, FUN = "prop",
+         legend = TRUE,
+         args.legend = list(x = "topright", bty = "n", inset=c(-0.32, 0)))
```

Figures 1 and 2 show potential informativeness in cluster size for the `screen8` data. Cluster size appears to be negatively associated with average standardized test scores but positively associated with the proportion of male students and the proportion participating in sports-related extracurricular activities. These empirical results can be verified using the test for ICS, implemented through the function `icstestClust`, as illustrated below. The result of this test suggests that cluster size is informative for standardized math test scores. Cluster size is also informative for standardized reading test scores, gender, and sports as an extracurricular activity ($p < .001$, results not shown).

```
R> set.seed(100)
R> ics.math <- icstestClust(screen8$math, screen8$sch.id, B = 1000,
+   print.it = FALSE)

R> ics.math
Test of informative cluster size (TF)
data: screen8$math
TF = 0.029686, p-value < 2.2e-16
```

Within the `icstestClust` function, the type of test statistic, TF or TCM as detailed earlier, is specified using the `test.method` argument, and the number of bootstrap loops by argument `B`. Argument `print.it` is a logical indicating whether to print the progress of the bootstrap procedure. We note that the need for bootstrap resampling in `icstestClust` can make its implementation computationally expensive.

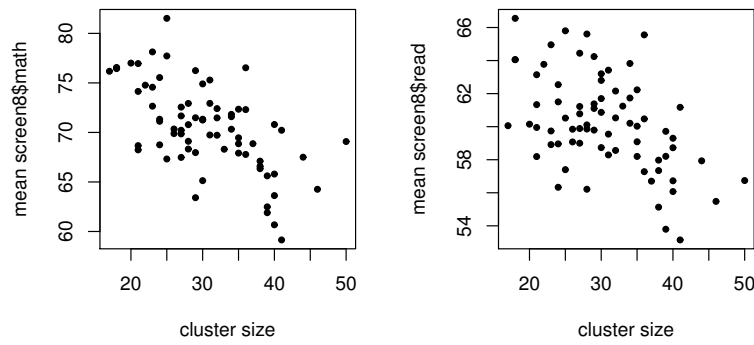


Figure 1: Average scores in maths and reading by cluster size in screen8 data.

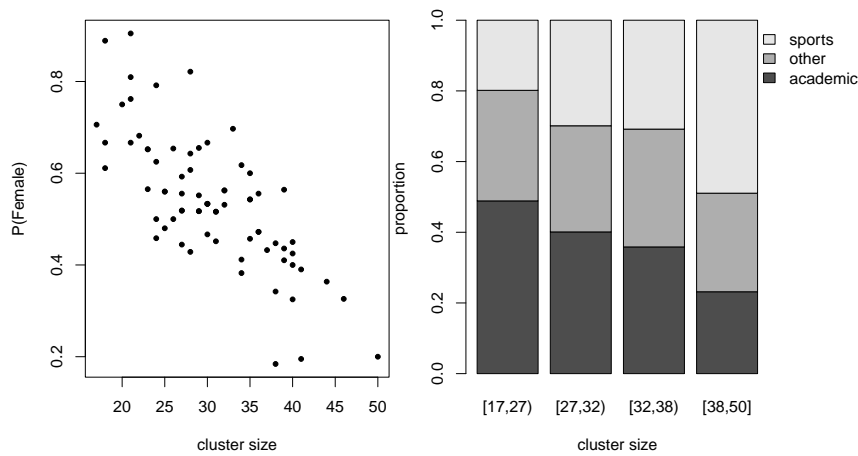


Figure 2: Plots of categorical variables by cluster size in screen8 data. Proportion of female students decreases with cluster size (left), whereas student participation in sports-related extracurricular activities increases with cluster size (right).

Testing a marginal proportion

The first question of interest suggests a one-sample test of a proportion via `proptestClust`. We specify a one-sided alternative and use the default sandwich variance estimator evaluated at the null value of the proportion (`variance = "sand.null"`), shown to perform best for this test (Gregg et al., 2020).

```
R> screen8$math.p <- 1*(screen8$math >= 65)
R> proptestClust(screen8$math.p, screen8$sch.id, p = .75, alternative = "great")
Cluster-weighted proportion test with variance est: sand.null

data: screen8$math.p, M = 73
z = 0.70159, p-value = 0.2415
alternative hypothesis: true p is greater than 0.75
95 percent confidence interval:
0.7311459 1.0000000
sample estimates:
Cluster-weighted proportion
0.7640235
```

As noted earlier, `htestClust` functions produce objects of class `htest`, producing familiar output through the `print` method for such objects. We conclude that the proportion of students with proficient math test scores is not greater than 0.75.

In the case that all clusters have a size of 1, the results of `htestClust` functions will be in general

correspondence with that of the classical analogue test, though exact results will differ slightly due to the reweighted tests relying on asymptotics. This is demonstrated through the following example.

```
R> set.seed(123)
R> x <- rbinom(100, size = 1, p = 0.7)
R> id <- 1:100
R> proptestClust(x, id)

Cluster-weighted proportion test with variance est: sand.null

data: x, M = 100
z = 4.2, p-value = 2.669e-05
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
0.6120018 0.8079982
sample estimates:
Cluster-weighted proportion
0.71

R> prop.test(sum(x), length(x))

1-sample proportions test with continuity correction

data: sum(x) out of length(x), null probability 0.5
X-squared = 16.81, df = 1, p-value = 4.132e-05
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
0.6093752 0.7942336
sample estimates:
p
0.71
```

Test of independence

The second question suggests a test of independence of extracurricular activity and gender. We start by producing cluster-weighted estimates of the proportion of students participating in each activity within each gender.

```
R> tab <- table(screen8$gender, screen8$activity, screen8$sch.id)
R> ptab <- prop.table(tab, c(1,3))
R> apply(ptab, c(1,2), mean)
academic other sports
F 0.3952102 0.2968473 0.3079425
M 0.3790267 0.3186699 0.3023035
```

The cluster-weighted proportions appear roughly similar, and we can test using `chisqtestClust`. Here, the default method of variance estimation is method of moments (`variance = "MoM"`), demonstrated to be best for the test of independence (Gregg et al., 2020).

```
R> chisqtestClust(screen8$gender, screen8$activity, screen8$sch.id)
Cluster-weighted Chi-squared test of independence with variance est:
MoM

data: screen8$gender and screen8$activity, M = 73
X-squared = 1.6131, df = 2, p-value = 0.4464
```

Before proceeding to the next analysis, we note that further evidence of ICS in the `screen8` data can be demonstrated by implementing the standard chi-squared test for this question, which suggests that females were more likely to participate in academic extracurricular activities and males in sports.

```
R> prop.table(table(screen8$gender, screen8$activity), 1)
academic other sports
F 0.3891323 0.2979842 0.3128834
M 0.3370268 0.3120960 0.3508772

R> chisq.test(screen8$gender, screen8$activity)
```

Pearson's Chi-squared test

```
data: screen8$gender and screen8$activity
X-squared = 6.9303, df = 2, p-value = 0.03127
```

Tests of quantitative variables for two or more groups

We compare math test scores between males and females using the `ttestClust` function. We conclude that mean standardized test scores are equivalent between males and females, a departure from the conclusion reached by the standard `t` test ($p < .001$, results not shown).

```
R> ttestClust(math ~ gender, id = sch.id, data = screen8)
```

Two sample group-weighted test of means

```
data: math by gender, M = 73
z = 1.3495, p-value = 0.1772
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.2234259  1.2111344
sample estimates:
weighted mean in group F weighted mean in group M
70.75124                70.25739
```

Even though this test does not make use of the t distribution, we have named it as such to parallel the standard t -test means (`t.test` in R). Multi-group tests of quantitative parameters in `htestClust` implement jackknife variance estimation, so specification of variance estimation method is not necessary. In addition to the formula implementation used above, we note that `ttestClust` can also accept vectors of data and cluster identifiers for each of the two groups.

An alternative approach to this comparison, particularly if test scores were skewed in any way, would be a rank-based test. `wilcoxtestClust` implements the group-weighted analogue of the Wilcoxon test, which we use as an alternative method for the comparison of math test scores between males and females.

```
R> wilcoxtestClust(math ~ gender, id = sch.id, data = screen8, method = "group")
Group-weighted rank sum test
```

```
data: math by gender, M = 73
z = -1.3799, p-value = 0.1676
alternative hypothesis: true location shift is not equal to 0
```

Our conclusion is the same as with the reweighted test of means. We note that this test requires estimation of the cluster-weighted empirical cumulative distribution (Dutta and Datta, 2016a) as well as jackknife variance estimation, so there is an added measure of computational expense in using `wilcoxtestClust`.

Finally, we compare reading test scores among the three groups defined by extracurricular activity, using `onewaytestClust`. Mean standardized reading test scores are not appreciably different among extracurricular activity groups.

```
R> onewaytestClust(read ~ activity, id = sch.id, data = screen8)
Reweighted one-way analysis of means for clustered data
```

```
data: read and activity, M = 73
X-squared = 1.3191, df = 2, p-value = 0.5171
sample estimates:
academic  other  sports
60.11498  60.40785  59.69659
```

We have not shown the full functionality of the above-demonstrated functions, nor the `htestClust` functions for testing correlation, marginal homogeneity, and variance listed in Table 1. Their syntax and usage is similar and fully documented with examples in the help files.

5 Discussion

Standard model-based inference of clustered data can be biased when cluster or group size is informative. Reweighting methods that correct for this bias have been established and a number of authors have applied such weighting to develop direct hypothesis tests of marginal parameters in clustered data. Such tests can be interpreted as clustered analogues to common classical statistical tests, and include methods related to ranks, correlation, proportions, means and variances. While these methods are effective and intuitive, all but a few of these tests have remained inaccessible to many researchers due to an absence of convenient software.

In this paper we introduced **htestClust**, which is the first R package designed as a comprehensive library of inferential methods appropriate for clustered data with ICS/IWCGS. Most functions in **htestClust** perform hypothesis tests for clustered data that have an analogous classical form, and the interface of the package has been designed to reflect this relationship. Function syntax has been purposefully structured to resemble that of functions available in the native R environment that perform the analogous classical tests. Many functions have been designed with a secondary interface that operates through table or formula input, allowing flexibility in data structure. In addition to the hypothesis tests of marginal parameters, **htestClust** also includes functions to visualize potential informativeness and test for ICS. These tools allow analysts to explore the effect and degree of informativeness in their data.

With the exception of the test for ICS, the hypothesis tests performed by **htestClust** are derived through the asymptotic normality of reweighted parameters, and their asymptotic convergence is indexed by the number of clusters. As such, their use should only be considered when the number of clusters is sufficiently large (at least 30). Additionally, these methods retain a cluster-based marginal interpretation, making them appropriate when clusters, rather than intra-cluster observations, are the unit of interest. The marginal nature of these tests provides researchers with an analysis corresponding to a snapshot in time. If analysis of temporal aspects or effects of additional covariates is desired, readers might instead consider reweighted model-based methods such as those by Bible et al. (2016), Neuhaus and McCulloch (2011), and Wang et al. (2011). Future research will also be devoted to developing tests adjusting for informativeness due to quantitative covariates measured at the individual-within-cluster level.

htestClust is a tool to facilitate the analysis of clustered data, and we have designed its use to be accessible and intuitive. While the inferential methods performed by this package have been developed to correct for the biasing effects of ICS/IWCGS, they remain applicable when fluctuations of cluster or group size are unrelated to the outcome of interest. As such, this package is an effective resource for researchers addressing marginal analyses in clustered data with any variation in the cluster and/or group sizes.

Computational details

The results in this paper were obtained using R 4.0.3 with the **MASS** 7.3.51 package.

Acknowledgments

The authors would like to thank Lucas Koepke and an anonymous reviewer for their thorough review and insightful comments that improved the quality of this manuscript.

Bibliography

- J. Bible, J. D. Beck, and S. Datta. Cluster adjusted regression for displaced subject data (cards): Marginal inference under potentially informative temporal cluster size profiles. *Biometrics*, 72(2):441–451, 2016. URL <https://doi.org/10.1111/biom.12456>. [p54, 64]
- V. J. Carey, T. Lumley, and B. Ripley. *gee: Generalized Estimation Equation Solver*, 2019. URL <https://CRAN.R-project.org/package=gee>. R package version 4.13-20. [p54]
- A. Chaurasia, D. Liu, and P. S. Albert. Pattern-mixture models with incomplete informative cluster size: Application to a repeated pregnancy study. *Journal of the Royal Statistical Society C*, 67(1):255, 2018. URL <https://doi.org/10.1111/rssc.12226>. [p54]
- S. Datta and G. A. Satten. Rank-sum tests for clustered data. *Journal of the American Statistical Association*, 100(471):908–915, 2005. URL <https://doi.org/10.1198/016214504000001583>. [p54, 55, 56]

- S. Datta and G. A. Satten. A signed-rank test for clustered data. *Biometrics*, 64(2):501–507, 2008. URL <https://doi.org/10.1111/j.1541-0420.2007.00923.x>. [p54, 55, 56]
- S. Dutta and S. Datta. A rank-sum test for clustered data when the number of subjects in a group within a cluster is informative. *Biometrics*, 72(2):432–440, 2016a. URL <https://doi.org/10.1111/biom.12447>. [p54, 55, 56, 63]
- S. Dutta and S. Datta. **ClusterRankTest**: *Rank Tests for Clustered Data*, 2016b. URL <https://CRAN.R-project.org/package=ClusterRankTest>. R package version 1.0. [p57]
- M. Gregg. *Marginal Methods and Software for Clustered Data With Cluster- and Group-Size Informativeness*. PhD thesis, UL (University of Louisville), 2020. [p54, 55, 56]
- M. Gregg, S. Datta, and D. Lorenz. Variance estimation in tests of clustered categorical data with informative cluster size. *Statistical Methods in Medical Research*, 29(11):3396–3408, 2020. URL <https://doi.org/10.1177/0962280220928572>. [p54, 55, 56, 61, 62]
- U. Halekoh, S. Højsgaard, J. Yan, et al. The r package **geepack** for generalized estimating equations. *Journal of Statistical Software*, 15(2):1–11, 2006. [p54]
- E. B. Hoffman, P. K. Sen, and C. R. Weinberg. Within-cluster resampling. *Biometrika*, 88(4):1121–1134, 2001. URL <https://doi.org/10.1093/biomet/88.4.1121>. [p55]
- Y. Huang and B. Leroux. Informative cluster sizes for subcluster-level covariates and weighted generalized estimating equations. *Biometrics*, 67(3):843–851, 2011. URL <https://doi.org/10.1111/j.1541-0420.2010.01542.x>. [p54, 56]
- A.-M. Iosif and A. R. Sampson. A model for repeated clustered data with informative cluster sizes. *Statistics in Medicine*, 33(5):738–759, 2014. URL <https://doi.org/10.1002/sim.5988>. [p54]
- Y. Jiang. *clusrank: Wilcoxon Rank Sum Test for Clustered Data*, 2018. URL <https://CRAN.R-project.org/package=clusrank>. R package version 0.6-2. [p57]
- D. J. Lorenz, S. Datta, and S. J. Harkema. Marginal association measures for clustered data. *Statistics in Medicine*, 30(27):3181–3191, 2011. URL <https://doi.org/10.1002/sim.4368>. [p54, 55, 56]
- D. J. Lorenz, S. Levy, and S. Datta. Inferring marginal association with paired and unpaired clustered data. *Statistical Methods in Medical Research*, 27(6):1806–1817, 2018. URL <https://doi.org/10.1177/0962280216669184>. [p54]
- A. Mitani, E. Kaye, and K. Nelson. Marginal analysis of multiple outcomes with informative cluster size. *Biometrics*, 2020. URL <https://doi.org/10.1111/biom.13241>. [p54]
- A. A. Mitani, E. K. Kaye, and K. P. Nelson. Marginal analysis of ordinal clustered longitudinal data with informative cluster size. *Biometrics*, 75(3):938–949, 2019. URL <https://doi.org/10.1111/biom.13050>. [p54]
- J. M. Neuhaus and C. E. McCulloch. Estimation of covariate effects in generalized linear mixed models with informative cluster sizes. *Biometrika*, 98(1):147–162, 2011. URL <https://doi.org/10.1093/biomet/asq066>. [p64]
- J. Nevalainen, H. Oja, and S. Datta. Tests for informative cluster size using a novel balanced bootstrap scheme. *Statistics in Medicine*, 36(16):2630–2640, 2017. URL <https://doi.org/10.1002/sim.7288>. [p55, 57]
- M. Pavlou. *Analysis of Clustered Data When the Cluster Size is Informative*. PhD thesis, UCL (University College London), 2012. [p54]
- M. Wang, M. Kong, and S. Datta. Inference for marginal linear models for clustered longitudinal data with potentially informative cluster sizes. *Statistical Methods in Medical Research*, 20(4):347–367, 2011. URL <https://doi.org/10.1177/0962280209347043>. [p64]
- J. M. Williamson, S. Datta, and G. A. Satten. Marginal analyses of clustered data when cluster size is informative. *Biometrics*, 59(1):36–42, 2003. URL <https://doi.org/10.1111/1541-0420.00005>. [p54, 55, 56]

Mary Gregg
Department of Bioinformatics and Biostatistics
University of Louisville
485 East Gray Street
Louisville, KY 40202, USA
ORCID: 0000-0003-2991-6939
mary.gregg@louisville.edu

Somnath Datta
Department of Biostatistics
University of Florida
2004 Mowry Rd
P.O. Box 117450
Gainesville, FL 32611 USA
ORCID: 0000-0003-4381-1842
somnath.datta@ufl.edu

Douglas Lorenz
Department of Bioinformatics and Biostatistics
University of Louisville
485 East Gray Street
Louisville, KY 40202, USA
ORCID: 0000-0001-8114-0926
djlore01@louisville.edu

akc: A Tidy Framework for Automatic Knowledge Classification in R

by Tian-Yuan Huang, Li Li, and Liying Yang

Abstract Knowledge classification is an extensive and practical approach in domain knowledge management. Automatically extracting and organizing knowledge from unstructured textual data is desirable and appealing in various circumstances. In this paper, the tidy framework for automatic knowledge classification supported by the **akc** package is introduced. With powerful support from the R ecosystem, the **akc** framework can handle multiple procedures in data science workflow, including text cleaning, keyword extraction, synonyms consolidation and data presentation. While focusing on bibliometric analysis, the **akc** package is extensible to be used in other contexts. This paper introduces the framework and its features in detail. Specific examples are given to guide the potential users and developers to participate in open science of text mining.

1 Introduction

Co-word analysis has long been used for knowledge discovery, especially in library and information science (Callon, Rip, and Law 1986). Based on co-occurrence relationships between words or phrases, this method could provide quantitative evidence of information linkages, mapping the association and evolution of knowledge over time. In conjunction with social network analysis (SNA), co-word analysis could be escalated and yield more informative results, such as topic popularity (Huang and Zhao 2019) and knowledge grouping (Khasseh et al. 2017). Meanwhile, in the area of network science, many community detection algorithms have been proposed to unveil the topological structure of the network (Fortunato 2010; Javed et al. 2018). These methods have then been incorporated into the co-word analysis, assisting to group components in the co-word network. Currently, the co-word analysis based on community detection is flourishing across various fields, including information science, social science and medical science (C.-P. Hu et al. 2013; J. Hu and Zhang 2015; Leung, Sun, and Bai 2017; Baziyad et al. 2019).

For implementation, interactive software applications, such as [CiteSpace](#) (Chen 2006) and [VOSviewer](#) (Van Eck and Waltman 2010), have provided freely available toolkits for automatic co-word analysis, making this technique even more popular. Interactive software applications are generally friendlier to users, but they might not be flexible enough for the whole data science workflow. In addition, the manual adjustments could be variant, bringing additional risks to the research reproducibility. In this paper, we have designed a flexible framework for automatic knowledge classification, and presented an open software package **akc** supported by R ecosystem for implementation. Based on community detection in co-occurrence network, the package could conduct unsupervised classification on the knowledge represented by extracted keywords. Moreover, the framework could handle tasks such as data cleaning and keyword merging in the upstream of data science workflow, whereas in the downstream it provides both summarized table and visualized figure of knowledge grouping. While the package was first designed for academic knowledge classification in bibliometric analysis, the framework is general to benefit a broader audience interested in text mining, network science and knowledge discovery.

2 Background

Classification could be identified as a meaningful clustering of experience, turning information into structured knowledge (Kwasnik 1999). In bibliometric research, this method has been frequently used to group domain knowledge represented by author keywords, usually listed as a part of co-word analysis, keyword analysis or knowledge mapping (He 1999; C.-P. Hu et al. 2013; Leung, Sun, and Bai 2017; Li, Ma, and Qu 2017; Wang and Chai 2018). While all named as (unsupervised) classification or clustering, the algorithm behind could vary widely. For instance, some researches have utilized hierarchical clustering to group keywords into different themes (J. Hu and Zhang 2015; Khasseh et al. 2017), whereas the studies applying VOSviewer have adopted a weighted variant of modularity-based clustering with a resolution parameter to identify smaller clusters (Van Eck and Waltman 2010). In the framework of **akc**, we have utilized the modularity-based clustering method known as community detection in network science (Newman 2004; Murata 2010). These functions are supported by the [igraph](#) package (Csardi, Nepusz, et al. 2006). Main detection algorithms implemented in **akc** include Edge betweenness (Girvan and Newman 2002), Fastgreedy (Clauset, Newman, and Moore 2004), Infomap (Rosvall and Bergstrom 2007; Rosvall, Axelsson, and Bergstrom 2009), Label propagation

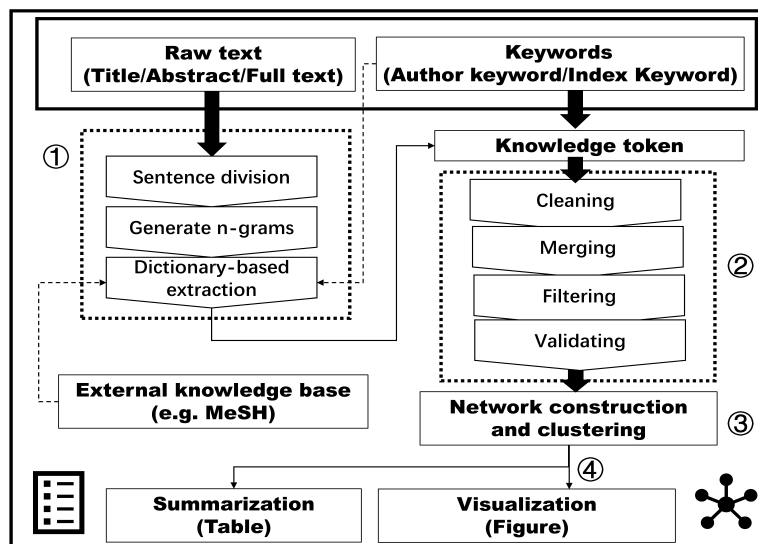


Figure 1: The design of akc framework. Generally, the framework includes four steps, namely: (1) Keyword extraction (optional); (2) Keyword preprocessing; (3) Network construction and clustering; (4) Results presentation.

(Raghavan, Albert, and Kumara 2007), Leading eigenvector (Newman 2006), Multilevel (Blondel et al. 2008), Spinglass (Reichardt and Bornholdt 2006) and Walktrap (Pons and Latapy 2005). The details of these algorithms and their comparisons have been discussed in the previous studies (Sousa and Zhao 2014; Yang, Algesheimer, and Tessone 2016; Garg and Rani 2017; Amrahov and Tugrul 2018).

In practical application, the classification result is susceptible to data variation. The upstream procedures, such as information retrieval, data cleaning and word sense disambiguation, play vital roles in automatic knowledge classification. For bibliometric analysis, the author keyword field provides a valuable source of scientific knowledge. It is a good representation of domain knowledge and could be used directly for analysis. In addition, such collections of keywords from papers published in specific fields could provide a professional dictionary for information retrieval, such as keyword extraction from raw text in the title, abstract and full text of literature. In addition to automatic knowledge classification based on community detection in keyword co-occurrence network, the **akc** framework also provides utilities for keyword-based knowledge retrieval, text cleaning, synonyms merging and data visualization in data science workflow. These tasks might have different requirements in specific backgrounds. Currently, **akc** concentrates on keyword-based bibliometric analysis of scientific literature. Nonetheless, the R ecosystem is versatile, and the popular tidy data framework is flexible enough to extend to various data science tasks from other different fields (Wickham et al. 2014; Wickham and Grolemund 2016; Silge and Robinson 2017), which benefits both end-users and software developers. In addition, when users have more specific needs in their tasks, they could easily seek other powerful facilities from the R community. For instance, **akc** provides functions to extract keywords using an n-grams model (utilizing facilities provided by **tidytext**), but skip-gram modelling is not supported currently. This functionality, on the other hand, could be provided in **tokenizers** (Mullen et al. 2018) or **quanteda** (Benoit et al. 2018) package in R. A greater picture of natural language processing (NLP) in R could be found in the [CRAN Task View: Natural Language Processing](#).

3 Framework

An overview of the framework is given in Figure 1. Note that the name **akc** refers to the overall framework for automatic keyword classification as well as the released R package in this paper. The whole workflow can be divided into four procedures: (1) Keyword extraction (optional); (2) Keyword preprocessing; (3) Network construction and clustering; (4) Results presentation.

(1) Keyword extraction (optional)

In bibliometric meta-data entries, the textual information of title, abstract and keyword are usually provided for each paper. If the keywords are used directly, there is no need to do information retrieval. Then we could directly skip this procedure and start from keyword preprocessing. However,

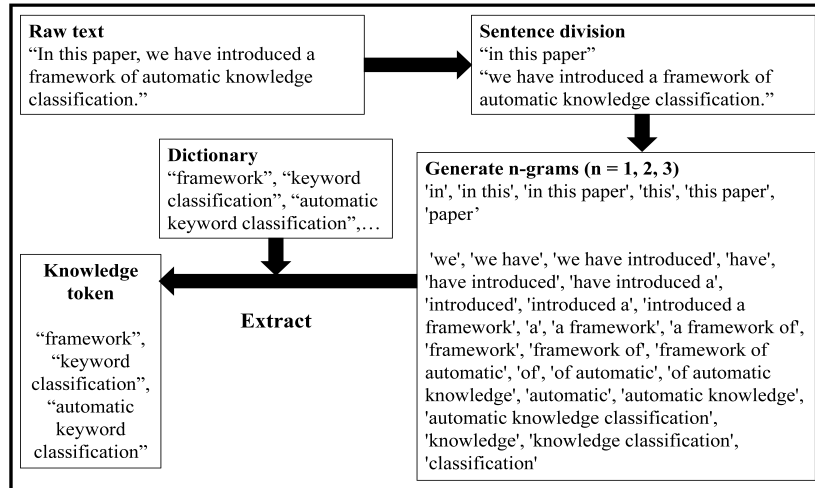


Figure 2: An example of keyword extraction procedure. The raw text would be first divided sentence by sentence, then tokenized to n-grams and yield the target keywords based on a dictionary. The letters are automatically turned to lower case.

Table 1: An example of keyword merging rule applied in akc. The keywords with the same lemma or stem would be merged to the highest frequency keyword in the original form.

ID	Original form	Lemmatized form	Merged form
1	higher education	high education	higher education
2	higher education	high education	higher education
3	high educations	high education	higher education
4	higher educations	high education	higher education
5	high education	high education	higher education
6	higher education	high education	higher education

sometimes the keyword field is missing, then we would need to extract the keywords from raw text in the title, abstract or full text with an external dictionary. At other times, one might want to get more keywords and their co-occurrence relationships from each entry. In such cases, the keyword field could serve as an internal dictionary for information retrieval in the provided raw text.

Figure 2 has displayed an example of keyword extraction procedure. First, the raw text would be split into sub-sentences (clauses), which suppresses the generation of cross-clause n-grams. Then the sub-sentences would be tokenized into n-grams. The n could be specified by the users, inspecting the average number of words in keyword phrases might help decide the maximum number of n. Finally, a filter is made. Only tokens that have emerged in the user-defined dictionary are retained for further analysis. The whole keyword extraction procedure could be implemented automatically with keyword_extract function in akc.

(2) Keyword preprocessing

In practice, the textualized contents are seldom clean enough to implement analysis directly. Therefore, the upstream data cleaning process is inevitable. In keyword preprocessing procedure of akc framework, the cleaning part would take care of some details in the preprocess, such as converting the letters to lower case and removing parentheses and contents inside (optional). For merging part, akc help merge the synonymous phrases according to their lemmas or stems. While using lemmatization and stemming might get abnormal knowledge tokens, here in akc we have designed a conversion rule to tackle this problem. We first get the lemmatized or stemmed form of keywords, then group them by their lemma or stem, and use the most frequent keyword in the group to represent the original keyword. This step could be realized by keyword_merge function in akc package. An example could be found in Table 1. After keyword merging, there might still be too many keywords included in the analysis, which poses a great burden for computation in the subsequent procedures. Therefore, a filter should be carried out here, it could exclude the infrequent terms, or extract top TF-IDF terms, or use any criteria that meets the need. Last, a manual validation should be carried out to ensure the final data quality.

(3) Network construction and clustering

Based on keyword co-occurrence relationship, the keyword pairs would form an edge list for construction of an undirected network. Then the facilities provided by the `igraph` package would automatically group the nodes (representing the keywords). This procedure could be achieved by using `keyword_group` function in `akc`.

(4) Results presentation

Currently, there are two kinds of output presented by `akc`. One is a summarized result, namely a table with group number and keyword collections (attached with frequency). Another is network visualization, which has two modes. The local mode provides a keyword co-occurrence network by group (use facets in `ggplot2`), whereas the global mode displays the whole network structure. Note that one might include a huge number of keywords and make a vast network, but for presentation the users could choose how many keywords from each group to be displayed. More details could be found in the following sections.

The `akc` framework could never be built without the powerful support provided by R community. The `akc` package was developed under R environment, and main packages imported to `akc` framework include `data.table` (Dowle and Srinivasan 2021) for high-performance computing, `dplyr` (Wickham et al. 2022) for tidy data manipulation, `ggplot2` (Wickham 2016) for data visualization, `ggraph` (Pedersen 2021) for network visualization, `ggwordcloud` (Le Pennec and Slowikowski 2019) for word cloud visualization, `igraph` (Csardi, Nepusz, et al. 2006) for network analysis, `stringr` (Wickham 2019) for string operations, `textstem` (Rinker 2018) for lemmatizing and stemming, `tidygraph` (Pedersen 2022) for network data manipulation and `tidytext` (Silge and Robinson 2016) for tidy tokenization. Getting more understandings on these R packages could help users utilize more alternative functions, so as to complete more specific and complex tasks. Hopefully, the users might also become potential developers of the `akc` framework in the future.

4 Example

This section shows how `akc` can be used in a real case. A collection of bibliometric data of *R Journal* from 2009 to 2021 is used in this example. The data of this example can be accessed in the [GitHub repository](#). Only the `akc` package is used in this workflow. First, we would load the package and import the data in the R environment.

```
library(akc)
rj_bib = readRDS("./rj_bib.rds")
rj_bib

#> # A tibble: 568 x 4
#>   id title                                     abstr~1 year
#>   <int> <chr>                                     <chr> <dbl>
#> 1     1 Aspects of the Social Organization and Trajectory of the~ Based ~ 2009
#> 2     2 asympTest: A Simple R Package for Classical Parametric S~ asympT~ 2009
#> 3     3 ConvergenceConcepts: An R Package to Investigate Various~ Conver~ 2009
#> 4     4 copas: An R package for Fitting the Copas Selection Model This a~ 2009
#> 5     5 Party on!                                Random~ 2009
#> 6     6 Rattle: A Data Mining GUI for R              Data m~ 2009
#> 7     7 sos: Searching Help Pages of R Packages      The so~ 2009
#> 8     8 The New R Help System                        Versio~ 2009
#> 9     9 Transitioning to R: Replicating SAS, Stata, and SUDAAN A~ Statis~ 2009
#> 10    10 Bayesian Estimation of the GARCH(1,1) Model with Student~ This n~ 2010
#> # ... with 558 more rows, and abbreviated variable name 1: abstract
```

`rj_bib` is a data frame with four columns, including `id` (Paper ID), `title` (Title of paper), `abstract` (Abstract of paper) and `year` (Publication year of paper). Papers in *R Journal* do not contain a keyword field, thus we have to extract the keywords from the title or abstract field (first step in Figure 1). Here in our case, we use the abstract field as our data source. In addition, we need a user-defined dictionary to extract the keywords, otherwise all the n-grams (meaningful or meaningless) would be extracted and the results would include redundant noise.

```
# import the user-defined dictionary
rj_user_dict = readRDS("./rj_user_dict.rds")
rj_user_dict
```

```

#> # A tibble: 627 x 1
#>   keyword
#>   <chr>
#> 1 seasonal-adjustment
#> 2 unit roots
#> 3 transformations
#> 4 decomposition
#> 5 combination
#> 6 integration
#> 7 competition
#> 8 regression
#> 9 accuracy
#> 10 symmetry
#> # ... with 617 more rows

```

Note that the dictionary should be a data.frame with only one column named “keyword”. The user can also use `make_dict` function to build the dictionary data.frame with a string vector. This function removes duplicated phrases, turns them to lower case and sorts them, which potentially improves the efficiency for the following processes.

```
rj_dict = make_dict (rj_user_dict$keyword)
```

With the bibliometric data (`rj_bib`) and dictionary data (`rj_dict`), we could start the workflow provided in Figure 1.

(1) Keyword extraction

In this step, we need a bibliometric data table with simply two informative columns, namely paper ID (*id*) and the raw text field (in our case *abstract*). The parameter *dict* is also specified to extract only keywords emerging in the user-defined dictionary. The implementation is very simple.

```

rj_extract_keywords = rj_bib %>%
  keyword_extract (id = "id", text = "abstract", dict = rj_dict)

```

By default, only phrases ranging 1 to 4 in length are included as extracted keywords. The user can change this range using parameters `n_min` and `n_max` in `keyword_extract` function. There is also a `stopword` parameter, allowing users to exclude specific keywords in the extracted phrases. The output of `keyword_extract` is a data.frame (tibble, `tbl_df` class provided by `tibble` package) with two columns, namely paper ID (*id*) and the extracted keyword (*keyword*).

(2) Keyword preprocessing

For the preprocessing part, `keyword_clean` and `keyword_merge` would be implemented in the cleaning part and merging part respectively. In the cleaning part, the `keyword_clean` function would: 1) Splits the text with separators (If no separators exist, skip); 2) Removes the contents in the parentheses (including the parentheses, optional); 3) Removes white spaces from start and end of string and reduces repeated white spaces inside a string; 4) Removes all the null character string and pure number sequences (optional); 5) Converts all letters to lower case; 6) Lemmatization (optional). The merging part has been illustrated in the previous section (see Table 1), thus would not be explained again. In the tidy workflow, the preprocessing is implemented via:

```

rj_cleaned_keywords = rj_extract_keywords %>%
  keyword_clean () %>%
  keyword_merge ()

```

No parameters are used in these functions because `akc` has been designed to input and output tibbles with consistent column names. If the users have data tables with different column names, specify them in arguments (`id` and `keyword`) provided by the functions. More details can be found in the help document (use `?keyword_clean` and `?keyword_merge` in the console).

(3) Network construction and clustering

To construct a keyword co-occurrence network, only a data table with two columns (with paper ID and keyword) is needed. All the details have been taken care of in the `keyword_group` function. However, the user could specify: 1) the community detection function (use `com_detect_fun` argument); 2) the filter rule of keywords according to frequency (use `top` or `min_freq` argument, or both). In our example, we would use the default settings (utilizing Fastgreedy algorithm, only top 200 keywords by frequency would be included).

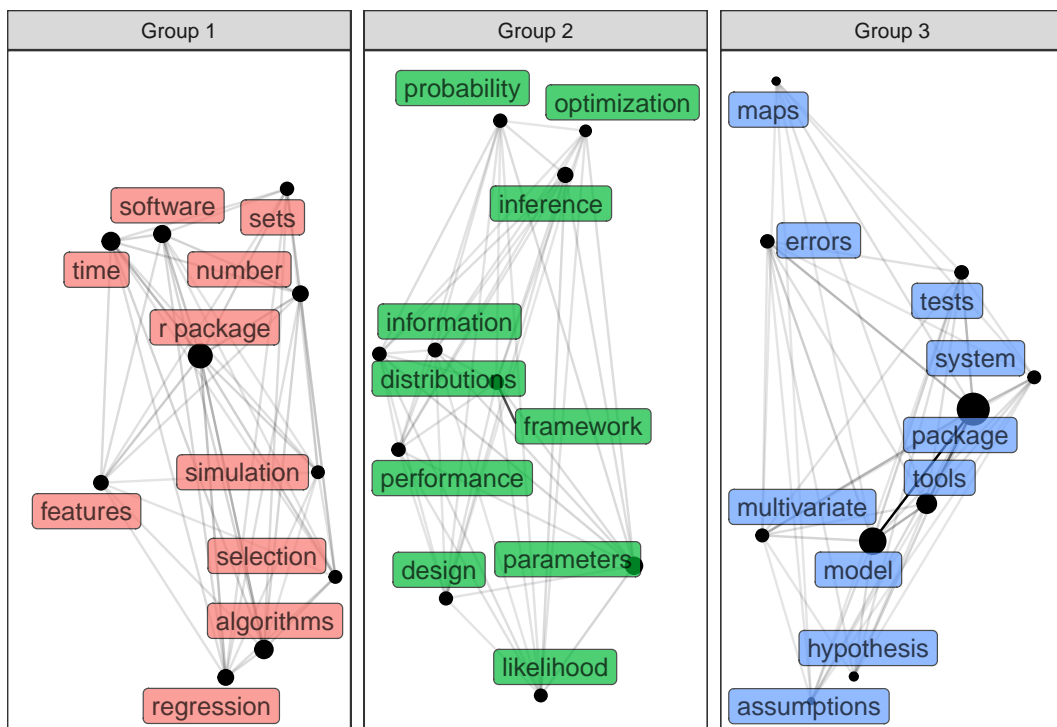


Figure 3: Network visualization for knowledge classification of R Journal (2009-2021). The keywords were automatically classified into three groups based on Fastgreedy algorithm. Only the top 10 keywords by frequency are displayed in each group.

```
rj_network = rj_cleaned_keywords %>%
  keyword_group ()
```

The output object `rj_network` is a `tbl_graph` class supported by [tidygraph](#), which is a tidy data format containing the network data. Based on this data, we can present the results in various forms in the next section.

(4) Results presentation

Currently, there are two major ways to display the classified results in [akc](#), namely network and table. A fast way to gain the network visualization is using `keyword_vis` function:

```
rj_network %>%
  keyword_vis ()
```

In Figure 3, the keyword co-occurrence network is clustered into three groups. The size of nodes is proportional to the keyword frequency, while the transparency degree of edges is proportional to the co-occurrence relationship between keywords. For each group, only the top 10 keywords by frequency are shown in each facet. If the user wants to dig into Group 1, `keyword_network` could be applied. Also, `max_nodes` parameter could be used to control how many nodes to be showed (in our case, we show 20 nodes in the visualization displayed in Figure 4).

```
rj_network %>%
  keyword_network (group_no = 1,max_nodes = 20)
```

Another displayed form is using table. This could be implemented by `keyword_table` via:

```
rj_table = rj_network %>%
  keyword_table ()
```

This would return a data.frame with two columns (see Table 2), namely the group number and the keywords (by default, only the top 10 keywords by frequency would be displayed, and the frequency information is attached).

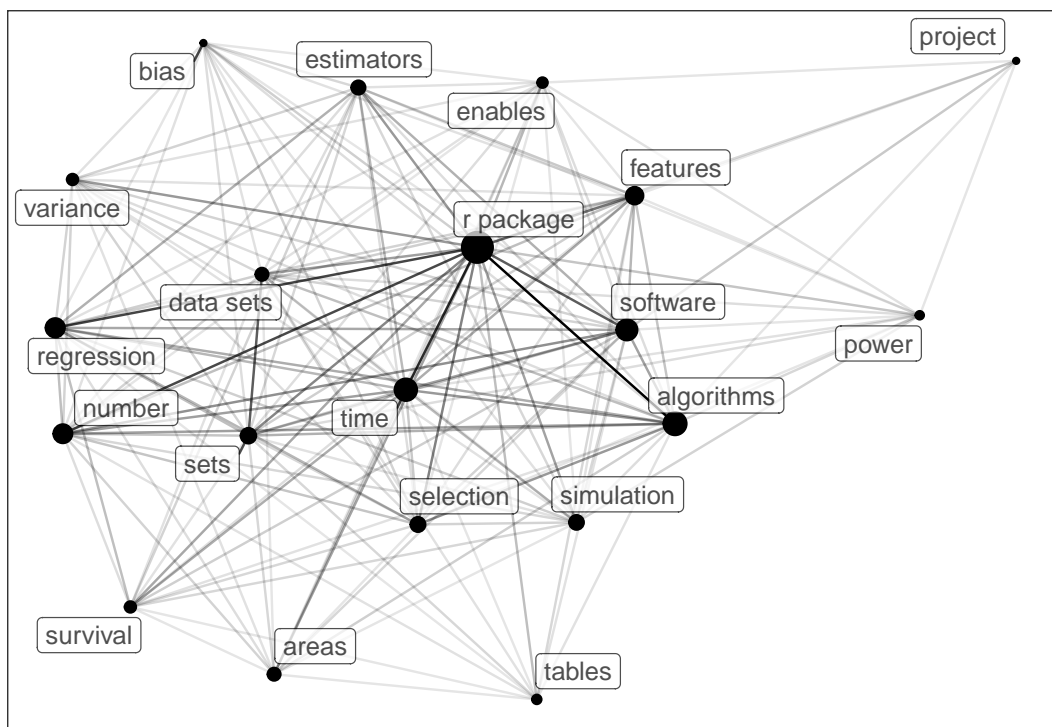


Figure 4: Focus on one cluster of the knowledge network of R journal (2009-2021). Top 20 keywords by frequency are shown in the displayed group.

Table 2: Top 10 keywords by frequency in each knowledge classification of R Journal (2009-2021).

Group	Keywords (TOP 10)
1	r package (238); algorithms (117); time (109); software (93); regression (75); number (72); features (60); sets (45); selection (41); simulation (40)
2	parameters (98); inference (65); framework (58); information (51); distributions (48); performance (47); probability (45); design (44); likelihood (41); optimization (31)
3	package (505); model (310); tools (140); tests (48); errors (46); multivariate (42); system (41); hypothesis (18); maps (16); assumptions (15)

Word cloud visualization is also supported by **akc** via **ggwordcloud** package, which could be implemented by using `keyword_ccloud` function.

In our example, we assume *R Journal* has a large focus on introducing R packages (Group 1 and Group 3 contains “r package” and “package” respectively). Common statistical subjects mentioned in *R Journal* include “regression” (in Group 1), “optimization” (in Group 2) and “multivariate” (in Group 3). While our example provides a preliminary analysis of knowledge classification in *R Journal*, an in-depth exploration could be carried out with a more professional dictionary containing more relevant keywords, and more preprocessing could be implemented according to application scenarios (e.g. “r package” and “package” could be merged into one keyword, and unigrams could be excluded if we consider them carrying indistinct information).

5 Discussion

The core functionality of the **akc** framework is to automatically group the knowledge pieces (keywords) using modularity-based clustering. Because this process is unsupervised, it can be difficult to evaluate the outcome of classification. Nevertheless, the default setting of community detection algorithm was selected after empirical tests via **benchmarking**. It was found that: 1) Edge betweenness and Spinglass algorithm are most time-consuming; 2) Edge betweenness and Walktrap algorithm could potentially find more local clusters in the network; 3) Label propagation could hardly divide the keywords into groups; 4) Infomap has high standard deviation of node number across groups. In the end, Fastgreedy was chosen as the default community detection algorithm in **akc**, because its performance is relatively stable, and the number of groups increases proportionally with the network size.

Though **akc** currently focuses on automatic knowledge classification based on community detection in keyword co-occurrence network, this framework is rather general in many natural language processing problems. One could utilize part of the framework to complete some specific tasks, such as word consolidating (using keyword merging) and n-gram tokenizing (using keyword extraction with a null dictionary), then export the tidy table and work in another environment. As long as the data follows the rule of tidy data format (Wickham et al. 2014; Silge and Robinson 2017), the **akc** framework could be easily decomposed and applied in various circumstances. For instance, by considering the nationalities of authors as keywords, **akc** framework could also investigate the international collaboration behavior in specific domain.

In the meantime, the **akc** framework is still in active development, trying new algorithms to carry out better unsupervised knowledge classification under the R environment. The expected new directions include more community detection functions, new clustering methods, better visualization settings, etc. Note that except for the topology-based community detection approach considering graph structure of the network, there is still another topic-based approach considering the textual information of the network nodes (Ding 2011), such as hierarchical clustering (Newman 2003), latent semantic analysis (Landauer, Foltz, and Laham 1998) and Latent Dirichlet Allocation (Blei, Ng, and Jordan 2003). These methods are also accessible in R, the relevant packages could be found in the [CRAN Task View: Natural Language Processing](#). With the tidy framework, **akc** could assimilate more nutrition from the modern R ecosystem, and move forward to create better reproducible open science schemes in the future.

6 Conclusion

In this paper, we have proposed a tidy framework of automatic knowledge classification supported by a collection of R packages integrated by **akc**. While focusing on data mining based on keyword co-occurrence network, the framework also supports other procedures in data science workflow, such as text cleaning, keyword extraction and consolidating synonyms. Though in the current stage it aims to support analysis in bibliometric research, the framework is quite flexible to extend to various tasks in other fields. Hopefully, this work could attract more participants from both R community and academia to get involved, so as to contribute to the flourishing open science in text mining.

7 Acknowledgement

This study is funded by The National Social Science Fund of China “Research on Semantic Evaluation System of Scientific Literature Driven by Big Data” (21&ZD329). The source code and data for reproducing this paper can be found at: https://github.com/hope-data-science/RJ_akc.

References

- Amrahov, S. E., and B. Tugrul. 2018. "A Community Detection Algorithm on Graph Data." In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 1–4.
- Baziyad, Hamed, Saeed Shirazi, Seyedmohammadreza Hosseini, and Rasoul Norouzi. 2019. "Mapping the Intellectual Structure of Epidemiology with Use of Co-Word Analysis." *Journal of Biostatistics and Epidemiology*.
- Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo. 2018. "quanteda: An R package for the quantitative analysis of textual data." *Journal of Open Source Software* 3 (30): 774.
- Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* 3: 993–1022.
- Blondel, Vincent D, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. "Fast Unfolding of Communities in Large Networks." *Journal of Statistical Mechanics: Theory and Experiment* 2008 (10): P10008.
- Callon, Michel, Arie Rip, and John Law. 1986. *Mapping the Dynamics of Science and Technology: Sociology of Science in the Real World*. Springer.
- Chen, Chaomei. 2006. "CiteSpace II: Detecting and visualizing emerging trends and transient patterns in scientific literature." *Journal of the American Society for Information Science and Technology* 57 (3): 359–77.
- Clauset, Aaron, M. E. J. Newman, and Cristopher Moore. 2004. "Finding Community Structure in Very Large Networks." *Physical Review E* 70 (December): 066111.
- Csardi, Gabor, Tamas Nepusz, et al. 2006. "The Igraph Software Package for Complex Network Research." *InterJournal, Complex Systems* 1695 (5): 1–9.
- Ding, Ying. 2011. "Community Detection: Topological Vs. Topical." *Journal of Informetrics* 5 (4): 498–514.
- Dowle, Matt, and Arun Srinivasan. 2021. *Data.table: Extension of 'Data.frame'*. <https://CRAN.R-project.org/package=data.table>.
- Fortunato, Santo. 2010. "Community Detection in Graphs." *Physics Reports* 486 (3): 75–174.
- Garg, Neha, and Rinkle Rani. 2017. "A comparative study of community detection algorithms using graphs and R." In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 273–78. IEEE.
- Girvan, Michelle, and Mark EJ Newman. 2002. "Community Structure in Social and Biological Networks." *Proceedings of the National Academy of Sciences* 99 (12): 7821–26.
- He, Q. 1999. "Knowledge Discovery Through Co-Word Analysis." *Library Trends* 48 (1): 133–59.
- Hu, Chang-Ping, Ji-Ming Hu, Sheng-Li Deng, and Yong Liu. 2013. "A co-word analysis of library and information science in China." *Scientometrics* 97 (2): 369–82.
- Hu, Jiming, and Yin Zhang. 2015. "Research patterns and trends of Recommendation System in China using co-word analysis." *Information Processing & Management* 51 (4): 329–39.
- Huang, Tian-Yuan, and Bin Zhao. 2019. "Measuring Popularity of Ecological Topics in a Temporal Dynamical Knowledge Network." *PloS One* 14 (1): e0208370.
- Javed, Muhammad Aqib, Muhammad Shahzad Younis, Siddique Latif, Junaid Qadir, and Adeel Baig. 2018. "Community detection in networks: A multidisciplinary review." *Journal of Network and Computer Applications* 108: 87–111.
- Khasseh, Ali Akbar, Faramarz Soheili, Hadi Sharif Moghaddam, and Afshin Mousavi Chelak. 2017. "Intellectual structure of knowledge in iMetrics: A co-word analysis." *Information Processing & Management* 53 (3): 705–20.
- Kwasnik, Barbara H. 1999. "The Role of Classification in Knowledge Representation and Discovery." *Library Trends* 48 (1): 22–47.
- Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. 1998. "An Introduction to Latent Semantic Analysis." *Discourse Processes* 25 (2-3): 259–84.
- Le Pennec, Erwan, and Kamil Slowikowski. 2019. *Ggwordcloud: A Word Cloud Geom for 'Ggplot2'*. <https://CRAN.R-project.org/package=ggwordcloud>.
- Leung, Xi Y, Jie Sun, and Billy Bai. 2017. "Bibliometrics of social media research: A co-citation and co-word analysis." *International Journal of Hospitality Management* 66: 35–45.
- Li, Xinjian, Emily Ma, and Hailin Qu. 2017. "Knowledge mapping of hospitality research- A visual analysis using CiteSpace." *International Journal of Hospitality Management* 60: 77–93.
- Mullen, Lincoln A., Kenneth Benoit, Os Keyes, Dmitry Selivanov, and Jeffrey Arnold. 2018. "Fast, Consistent Tokenization of Natural Language Text." *Journal of Open Source Software* 3 (23): 655.
- Murata, Tsuyoshi. 2010. "Detecting Communities in Social Networks." In *Handbook of Social Network Technologies and Applications*, edited by Borko Furht, 269–80. Boston, MA: Springer US.
- Newman, Mark EJ. 2003. "The Structure and Function of Complex Networks." *SIAM Review* 45 (2): 167–256.
- . 2004. "Fast Algorithm for Detecting Community Structure in Networks." *Physical Review E* 69

- (6): 066133.
- . 2006. "Finding Community Structure in Networks Using the Eigenvectors of Matrices." *Physical Review E* 74 (3): 036104.
- Pedersen, Thomas Lin. 2021. *Ggraph: An Implementation of Grammar of Graphics for Graphs and Networks*. <https://CRAN.R-project.org/package=ggraph>.
- . 2022. *Tidygraph: A Tidy API for Graph Manipulation*. <https://CRAN.R-project.org/package=tidygraph>.
- Pons, Pascal, and Matthieu Latapy. 2005. "Computing Communities in Large Networks Using Random Walks." In *International Symposium on Computer and Information Sciences*, 284–93. Springer.
- Raghavan, Usha Nandini, Reka Albert, and Soundar Kumara. 2007. "Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks." *Physical Review E* 76 (3): 036106.
- Reichardt, Jorg, and Stefan Bornholdt. 2006. "Statistical Mechanics of Community Detection." *Physical Review E* 74 (1): 016110.
- Rinker, Tyler W. 2018. *textstem: Tools for Stemming and Lemmatizing Text*. Buffalo, New York. <http://github.com/trinker/textstem>.
- Rosvall, Martin, Daniel Axelsson, and Carl T Bergstrom. 2009. "The Map Equation." *The European Physical Journal Special Topics* 178 (1): 13–23.
- Rosvall, Martin, and Carl T Bergstrom. 2007. "An Information-Theoretic Framework for Resolving Community Structure in Complex Networks." *Proceedings of the National Academy of Sciences* 104 (18): 7327–31.
- Silge, Julia, and David Robinson. 2016. "tidytext: Text Mining and Analysis Using Tidy Data Principles in R." *Journal of Open Source Software* 1 (3). <https://doi.org/10.21105/joss.00037>.
- . 2017. *Text Mining with r: A Tidy Approach*. O'Reilly Media, Inc.
- Sousa, Fabiano Berardo de, and Liang Zhao. 2014. "Evaluating and Comparing the Igraph Community Detection Algorithms." In *2014 Brazilian Conference on Intelligent Systems*, 408–13. IEEE.
- Van Eck, Nees Jan, and Ludo Waltman. 2010. "Software Survey: VOSviewer, a Computer Program for Bibliometric Mapping." *Scientometrics* 84 (2): 523–38.
- Wang, Mengyang, and Lihe Chai. 2018. "Three New Bibliometric Indicators/Approaches Derived from Keyword Analysis." *Scientometrics* 116 (2): 721–50.
- Wickham, Hadley et al. 2014. "Tidy Data." *Journal of Statistical Software* 59 (10): 1–23.
- Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.
- . 2019. *Stringr: Simple, Consistent Wrappers for Common String Operations*. <https://CRAN.R-project.org/package=stringr>.
- Wickham, Hadley, Romain Francois, Lionel Henry, Kirill Muller, et al. 2022. *Dplyr: A Grammar of Data Manipulation*. <https://CRAN.R-project.org/package=dplyr>.
- Wickham, Hadley, and Garrett Grolemund. 2016. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc.
- Yang, Zhao, Rene Algesheimer, and Claudio J Tessone. 2016. "A Comparative Analysis of Community Detection Algorithms on Artificial Networks." *Scientific Reports* 6 (1): 1–18.

Tian-Yuan Huang
National Science Library, Chinese Academy of Sciences
Beijing, China
ORCID: 0000-0002-4151-3764
huangtianyuan@mail.las.ac.cn

Li Li
National Science Library, Chinese Academy of Sciences; Department of Library, Information and Archives Management, School of Economics and Management, University of Chinese Academy of Science
Beijing, China
lili2020@mail.las.ac.cn

Liyang Yang
National Science Library, Chinese Academy of Sciences
Beijing, China
yangly@mail.las.ac.cn

APCI: An R and Stata Package for Visualizing and Analyzing Age-Period-Cohort Data

by Jiahui Xu, Liying Luo

Abstract Social scientists have frequently attempted to assess the relative contribution of age, period, and cohort variables to the overall trend in an outcome. We develop an R package **APCI** (and Stata command `apci`) to implement the age-period-cohort-interaction (APC-I) model for estimating and testing age, period, and cohort patterns in various types of outcomes for pooled cross-sectional data and multi-cohort panel data. Package **APCI** also provides a set of functions for visualizing the data and modeling results. We demonstrate the usage of package **APCI** with empirical data from the Current Population Survey. We show that package **APCI** provides useful visualization and analytical tools for understanding age, period, and cohort trends in various types of outcomes.

1 Introduction

Researchers across disciplines have long been interested in distinguishing the relative contribution of three time-related variables — namely, age (i.e., how old a person is at the time of data collection), time periods (e.g., the Great Recession 2007-2009 and the COVID-19 pandemic beginning in December 2019), and cohort membership (e.g., the baby boom cohort born in 1945-1964 and the Millennials born in 1981-1996) — to the overall trends in various outcomes (e.g., labor force participation, attitudes, and cognitive functioning) (Alwin and McCammon, 2003; Clogg, 1982; Pescosolido et al., 2021). Decomposing the overall trends into age, period, and cohort variations provides insight into the ways in which biological and social factors affect these outcomes (Hobcraft et al., 1982; Heckman and Robb, 1985; Fosse and Winship, 2019).

To quantify the relative contribution of age, period, and cohort, Luo and Hodges (2020a) have recently developed a model called the age-period-cohort-interaction (APC-I) model. The APC-I is qualitatively different from other age-period-cohort (APC) models in that it characterizes cohort effects as a structure of the age-by-period interaction terms to acknowledge the interdependence of age, period, and cohort effects, whereas prior methods attempt to recover the independent and additive effects of the three variables. The APC-I model has been used to understand the unique contribution of cohort membership in various outcomes including crime involvement, substance use, and cultural taste (Lu and Luo, 2020; Verdery et al., 2020; Ma, 2020). However, the authors of the APC-I model focused on the conceptual motivation of the method and offered relatively few technical details for implementing the method. Estimating and testing cohort effects in the APC-I model may be challenging for interested readers.

We developed an R package **APCI** (Xu and Luo, 2021) and a Stata command `apci` for implementing the APC-I model in empirical research using pooled cross-sectional data (e.g., the General Social Survey and the Current Population Survey) and importantly, extend the APC-I method for analyzing multi-cohort longitudinal or panel data (e.g., data from the Health and Retirement Study (HRS) and the National Longitudinal Study of Youth (NLSY)). The purpose of this paper is three folded. First, we describe the R functions in the **APCI** package and Stata command to estimate and test age, period, and cohort effects in the APC-I model. The core function can be used for analyzing pooled cross-sectional data and multi-cohort longitudinal data. Second, we introduce a set of visualization tools to help researchers motivate an APC analysis and interpret age, period, and cohort effects from the APC-I model. Third, we clarify several important issues about characterizing cohort effects as a set of age-by-period interaction terms. We pay particular attention to the implications of coding schemes and how to interpret the between-cohort average deviations and within-cohort life-course variations.

This paper is organized as follows. Following a description of traditional APC models and the identification problem, we introduce the APC-I model and the estimation and testing procedures. We explain how and why the age-by-period interaction terms can be used to characterize cohort effects with particular attention to the implications of coding schemes for estimating and testing interactions. Next, we describe the visualization tools and functions in the R package **APCI**. We then demonstrate how to use the package with the empirical example of men's and women's labor force participation from 1990 to 2018 in the United States using data from the Current Population Survey (CPS, Flood et al., 2021).

2 Methodology: the APC-I model

The APC identification problem

To formally estimate and infer the independent age, period, and cohort effects, [Mason et al. \(1973\)](#) specified an analysis of variance (ANOVA) model that they labeled the age-period-cohort (APC) accounting model:

$$g\left(E\left(Y_{ij}\right)\right)=\mu+\alpha_i+\beta_j+\gamma_k \quad (1)$$

for age groups $i=1,2,\dots,A$, periods $j=1,2,\dots,P$, and cohorts $k=1,2,\dots,(A+P-1)$, where $\sum_{i=1}^A \alpha_i = \sum_{j=1}^P \beta_j = \sum_{k=1}^{A+P-1} \gamma_k = 0$. $E\left(Y_{ij}\right)$ denotes the expected value of the outcome Y for the i th age group in the j th time period; g is the “link function”; α_i denotes the mean difference from the global mean μ associated with the i th age category; β_j denotes the mean difference from μ associated with the j th period; γ_k denotes the mean difference from μ associated with membership in the k th cohort.

Unfortunately, the APC accounting model (1) is not identified even when a coding scheme (e.g., dummy coding where one group is set as the reference group or effect coding where the sum of the coefficients for each effect is set to 0) is applied. This is because age, period, and cohort are exactly linearly related (see [Fienberg and Mason, 1979](#); [Fosse and Winship, 2019](#); [Luo et al., 2016](#), for detailed discussions). As a result, the design matrix of model (1) has rank one less than full, so an infinite number of solutions (i.e., estimates) for the parameters fit the data equally well. That is, the data cannot distinguish different estimation results, so an additional constraint — in addition to the usual reference group or sum-to-zero constraint — must be imposed in order to choose one set of estimates. Moreover, interpreting the results is difficult because the standard interpretation of regression coefficients — that is, the conditional effect of one variable after accounting for other covariates — cannot apply due to the lack of variation in the third variable (e.g., cohort) after considering the other two (e.g., age and period).

The theoretical root of the identification problem in traditional APC models is the problematic assumption that age, period, and cohort effects operate independently of each other. It implies that the identification challenge is inherent in any APC model that attempts to separate independent and additive effects of age, period, and cohort and thus cannot be solved by changing the model setup (e.g., using random effects for period and cohort as in [Yang and Land, 2006](#); see [Luo and Hodges, 2020b](#), for a critique) or by variable manipulation (e.g., using unequal interval widths for age, period, and cohort groups as in [Robertson and Boyle, 1986](#); [Sarma et al., 2012](#); see [Luo et al., 2016](#), for a detailed discussion). The identification problem is well recognized, and its consequences have been discussed extensively ([Fienberg and Mason, 1985](#); [Fosse and Winship, 2019](#); [Kupper et al., 1983, 1985](#); [Luo et al., 2016](#); [Luo and Hodges, 2020b](#); [te Grotenhuis et al., 2016](#); [O’Brien, 2020](#); [Morgan and Lee, 2021](#); [Luo, 2013](#)). In essence, internal information derived from the data cannot help because the problem is circular: researchers do the analysis to learn precisely the kind of information needed to justify any such constraint.

The APC-I model

[Luo and Hodges \(2020a\)](#) proposed a new APC model called the age-period-cohort-interaction (APC-I) model. The APC-I model is qualitatively different from all estimators developed under the traditional framework in that it explicitly specifies cohort effects as a structure of the age-by-period interactions. A life-course dynamics hypothesis that concerns about whether and how cohort effects may change as cohorts age thus corresponds to a specific structure of the age-by-period interactions. This specification is motivated by the theoretical account that “The minimal basis for expecting interdependence between inter-cohort differentiation and social change is that change has variant import for persons of unlike age” ([Ryder, 1965](#)). That is, a basic notion on which cohort analysis rests is that “transformations of the social world modify people of different ages in different ways.” ([Ryder, 1965](#))

The APC-I model is fully identified in the sense that it does not require additional constraints other than a regular coding scheme. It is also flexible enough to test various hypotheses about life-course changes within cohorts as cohort members age. We first describe the model specification and estimation and testing techniques. The next section demonstrates the procedure using empirical examples.

The general form of the APC-I model can be written as:

$$g\left(E\left(Y_{ij}\right)\right)=\mu+\alpha_i+\beta_j+\alpha\beta_{ij(k)} \quad (2)$$

where g , Y_{ij} , μ , α_i and β_j are defined as in model (1) and $\alpha\beta_{ij(k)}$ denotes the interaction of the i th age group and j th period group, corresponding to the effect of the k th cohort. Note that except for the oldest and youngest cohorts, the effect of one cohort includes multiple age-by-period interaction terms $\alpha\beta_{ij(k)}$ that lie on the same diagonal in a table with ages in rows and periods in columns.

Model (2) differs from model (1) in the way that cohort effects are modeled. In model (2), cohort effects are considered as a specific form of the age-by-period interaction. In statistics, the interaction between two variables describes the differential effects of one variable depending on the level of the other (Scheffé, 1999). In APC research, this means that if part of the overall pattern of interest can be attributed to cohort differences, significant age-by-period interactions should be present. When cohort membership is not associated with the outcome — that is, when the effects of historical or social shifts (period effects) are uniform across age groups — then age-by-period interactions should not be present.

Luo and Hodges (2020a) described a procedure for investigating age and period main effects and inter-cohort deviations and intra-cohort dynamics. They recommended beginning with a deviance test about whether the effects of time periods vary among age groups, which is called “a global F test”. A non-significant global F statistic indicates that there are few age-by-period interaction effects and thus little cohort variation. If the model suggests significant age-by-period interaction effects, one may proceed to examine inter- and intra-cohort differences¹. Inter-cohort average deviations are quantified based on the arithmetic mean of the age-by-period interaction terms contained in each cohort and a t test can be used to examine the average of that cohort-specific deviation. To investigate intra-cohort dynamics over the life course (e.g., the cumulative (dis)advantage hypothesis in Dannefer, 1987; Ferraro and Morton, 2018; Chauvel et al., 2016; O’Brien, 2020), one may use a t-test of the linear orthogonal polynomial contrast in each cohort’s age-by-period interaction effects. This intra-cohort life-course dynamics test is helpful for investigating whether the average (dis)advantages of the members of that cohort accumulate, remain stable, or diminish as they age.

The APC-I model has three advantages. First, it is identified in that it does not require additional constraints other than the usual coding scheme. That is, it avoids the identification problem of the APC accounting model based on the theoretical account of cohort effects and allows inclusion of other important predictors such as education, sex, and employment status. Second, the interpretation of the coefficient estimates of the APC-I model is meaningful and straightforward. This is because the APC-I model recognizes the dependence of age, period, and cohort so the dilemma that analysts face using traditional APC models does not apply. Third, the APC-I model permits investigating life-course dynamics as a cohort ages, whereas extant methods usually assume that cohort effects do not change.

It is important to note that the APC-I model is never intended to “solve” the identification problem in traditional APC accounting models because it is a false problem to begin with. Given the near monopoly of the accounting model, it may be challenging not to see the APC-I method through the lens of the traditional APC accounting framework. For example, because the APC-I model quantifies cohort effects as a structure of the age-by-period interactions, some readers may take it to mean that the APC-I model cannot estimate “linear cohort main effects”. However, the APC-I method, by design, does not intend to estimate any kind of “linear cohort main effects” precisely because the traditional model’s assumption that there is a linear cohort effect that is additive or independent of age and period effects lacks theoretical grounding and is thus arbitrary and questionable. Please see Luo and Hodges (2020a) for a more thorough discussion about the theoretical motivation of the APC-I model.

Because the APC-I mode is relatively new, below we make additional remarks about interaction effects and coding schemes to help readers better understand and use the model.

Interaction effects

In some cases, interaction terms may be difficult to interpret besides suggesting that the effect of one variable may depend on the values of the other. However, as explained by Luo and Hodges (2020a), the age-by-period interaction terms correspond to the conceptual definition of cohort effects and thus can be modeled and interpreted in a meaningful way. Specifically, cohort effects are expected when the influence of social events and changes differ by age groups. This conceptualization of cohort effects implies that the age-by-period interactions, which represent the differential effects of time periods depending on age, can be used to measure cohort effects.

Technically, because of the linear dependency among age, period, and cohort, the effects of the third variable can be expressed as the interaction between the other two variables. The APC-I model considers age and period as main effects and cohort their interactions, which may give the impression that it privileges age and period effects and “discriminates” against cohort effects. The theoretical motivation for this choice is that it is often desirable to estimate a general age pattern that individuals

¹The local deviation test is unavailable in the current version (1.0.5) of the R package that we develop.

follow as they get older. Period main effects are used to represent the kind of impacts of social environment that everyone in the society is exposed to. The decision to quantify cohort effects as a specific form of age-by-period interaction is informed by the demographic literature on how cohort effects are conceptualized in relation to age and period effects. Empirically, as the analysis of women's labor force participation in section [Examples](#) will illustrate, the size of the cohort effects, characterized as the age-by-period interactions, is not necessarily smaller—in fact may be larger—than some of the main effects.

Coding scheme and contrast

For the unidentified APC accounting model, some estimation methods including the intrinsic estimator yield effect estimates that are dependent on the choice of coding schemes in that estimates under different coding schemes are not equivalent (see [Luo et al., 2016](#); [te Grotenhuis et al., 2016](#), for a more detailed discussion). The APC-I model does not have a rank deficiency problem in the sense that it does not require more constraints than a usual ANOVA model with main effects and their interactions. For any identified model including the APC-I model, the estimates are equivalent; that is, the estimated cell means are the same for all coding schemes.

Although this equivalence holds for both main effects and interaction estimates in the APC-I model, it is less obvious for interaction terms because the interpretation of the specific parameter estimates do change with coding schemes. To illustrate, consider an example of applying the APC-I model to health data with three age categories and three periods, shown in [table 1](#) below. Under dummy coding—for example, the youngest age group 20-24 and the beginning survey period of 2000 are set to zero or omitted as the referent—the interaction for ages 25-29 and period 2005 in cell Y represents the difference in a health outcome between periods 2000 and 2005 for age 25-29 or equivalently, health difference between ages 25-29 and 20-24 for the period 2005. That is, interactions under dummy coding represents a directional difference from a particular reference group.

		Period		
		2000	2005	2010
Age	20 – 24			
	25 – 29		Y	
	30 – 34			

Table 1: Hypothetical data with three age categories and three periods illustrating a shift in meaning and interpretation of interaction terms under different coding schemes. The interaction terms under different types of coding in cell Y necessarily have different numerical values because of different reference groups. For example, the interaction term in cell Y under dummy coding represents a directional difference from a particular reference group (e.g., age 20-24 in year 2000). Under effect coding, the same interaction term in cell Y represents the deviation in the outcome from the age main effect plus period main effect for the group of individuals who were age 25-29 and surveyed in period 2005. Such different numeric values do not arise from an identification problem but rather from a shift in what these quantities represent.

By contrast, under effect coding (i.e., sum-to-zero coding), the same interaction term in cell Y represents the deviation in the health outcome from the age main effect plus period main effect for the group of individuals who were age 25-29 and surveyed in period 2005.

The estimated interaction terms under these two types of coding in cell Y thus necessarily have different numerical values. However, this difference does not arise from an identification problem but rather from a shift in what these quantities represent. That is, the two interaction terms can be transformed to be equivalent so that the means in Y after considering age and period main effects are the same under the two coding schemes.

We recommend using effect or sum-to-zero coding for estimating the APC-I model for the following reasons: when characterizing cohort effects as a set of age-by-period interactions, we are less concerned about any direction of the interactions; that is, we are not particularly interested in the difference between two cohorts at a particular age or time period. Rather, we focus on particular structures of these interactions that may represent theoretically interesting patterns during a cohort's life span. Effect coding is helpful because they all have the same referent group — the next lower level in the hierarchy of main effects and interactions. That is, we choose effect coding for the purpose of easy interpretation. This is also consistent with the recommendation of coding schemes in the presence of interactions ([Aiken et al., 1991](#); [Jaccard and Turrissi, 2003](#)).

3 The R package APCI

Package installation

The R package [APCI](#)² can be installed and loaded using the following R code³:

```
# install R package APCI
> install.packages("APCI")
# load R package APCI to the current working environment
> library(APCI)
```

The main routines to implement the APC-I model using package [APCI](#) are `apci.plot.raw`, `apci`, `apci.plot` (or `apci.plot.hexagram`, `apci.plot.heatmap`). A summary of these functions and input arguments used in the routines are described below.

Functions in R package APCI

The R package [APCI](#) contains the following functions for estimating the APC-I model and visualizing the data and the model results:

- `apci`: to estimate the age, period, and cohort effects using the APC-I model.
- `temp_model`: an internal function that estimates a generalized linear model.
- `tests`: to conduct the global F test.
- `maineffect`: an internal function to extract age and period main effects.
- `cohortdeviation`: an internal function to extract between-cohort average deviations and within-cohort life-course dynamics.
- `ageperiod_group`: to return a cohort index based on how age and period are grouped.
- `apci.plot.raw`: to visualize the mean values of the outcome across age and period groups, respectively.
- `apci.plot.hexagram`: to visualize the estimated cohort effects in a hexagram style.
- `apci.plot.heatmap`: to visualize the estimated cohort effects in a heatmap style.
- `apci.plot`: to visualize the estimated age, period, and cohorts in conventional figures.

A summary of input arguments required in these functions will be given one by one⁴ in the next section. Package [APCI](#) also contains three empirical datasets *women9017*, *cpsmen*, *cpswomen*, and one simulated dataset *simulation*. Dataset *women9017* was used and described in [Luo and Hodges \(2020a\)](#). Applications of the APC-I model to the other two empirical datasets are given in section [Examples](#).

Function apci

Function `apci` is the core function in the R package [APCI](#). It fits an APC-I model with or without covariates and returns a list of results including coefficients and standard error estimates for age main effects, period main effects, inter-cohort average deviations, and intra-cohort life-course trends, and covariate coefficients if any. Both pooled cross-sectional data and multi-cohort longitudinal/panel data are supported. Specifically, function `apci` is used as

```
apci(data, outcome, age, period, cohort, weight, covariate, family,
      dev.test=TRUE, print, gee, id, corstr,...)
```

and takes the following arguments:

- `data`: a data frame containing an outcome variable, age group indicators, period group indicators, and covariates to be used in the model. If a variable is not found in data, there will be an error message reminding users to check the input data again. Supported data structures include pooled cross-sectional data and multi-cohort longitudinal/panel data.

²The R package [APCI](#) works well in R version above 3.6.0., but updating to the latest version of R is highly recommended.

³If users have never installed packages in R or RStudio, use the following R code instead: `install.packages("APCI", repos = "http://cran.us.r-project.org")`.

⁴Summaries for internal functions are not listed. Please see [APCI reference manual](#) for details about internal functions.

- `outcome`: an object of class character containing the name of the outcome variable. The outcome variable can be a continuous, categorical, or count variable.
- `age`: an object of class character indicating the age group index taking on the number of distinct values in the data (e.g., six age groups: 20-24, 25-29, 30-34, 35-39, 40-44, and 45-49). The vector should be a factor (or “category”, or “enumerated type”).
- `period`: an object of class character indicating the time period index in the data.
- `cohort`: an optional object of class character indicating cohort membership index in the data. The cohort index can be generated from the age group index and time period index in the data because of the exact linear relationship among these three time-related indices.
- `weight`: an optional vector of sample weights to be used in the model fitting process. If non-NULL, user-supplied weights will be used in the first step to estimate the model. Observations with negative weights will be dropped in modeling.
- `covariates`: an optional vector of characters containing the names of user-specified covariate(s) to be used in the model. If the variables are not found in data, there will be an error message reminding the users to check the data again.
- `dev.test`: logical, specifying if the global F test (step 1) should be implemented before fitting the APC-I model. If TRUE, `apci` will first run the global F test and report the test results; otherwise, `apci` will skip this step and return NULL. The default setting is TRUE. However, users should be aware that the algorithm will not automatically stop even if there is no significant age-by-period interactions based on the global F test.⁵
- `family`: a character string specifying the link function to be used in the model. The value can be “binomial”, “multinomial”, or “gaussian”. See R function `glm` for more details about link functions.
- `print`: logical, specifying if the intermediate results should be displayed in the console when fitting the model. The default setting is TRUE to display the results of each procedure.
- `gee`: logical, indicating if the data is cross-sectional data or longitudinal/panel data. If TRUE, the generalized estimating equation will be used to correct the standard error estimates. The default is FALSE, indicating that the data are cross-sectional.
- `id`: a character vector specifying the cluster index in longitudinal data. It is required when `gee` is TRUE. The length of the vector should be the same as the number of observations.
- `corstr`: a character string specifying a possible correlation structure in the error terms when `gee` is TRUE. The following are allowed: `independence`, `fixed`, `stat_M_dep`, `non_stat_M_dep`, `exchangeable`, `AR-M` and `unstructured`. The default value is `exchangeable`.
- `unequal_interval`: logical, indicating if age and period groups are of the same interval width. The default is set as TRUE.
- `age_range`, `period_range`: numeric vectors indicating the actual age or period range (e.g., 10 to 59 years old or from 2000 to 2019).
- `age_interval`, `period_interval`, `age_group`, `period_group`: numeric values or character vectors indicating how age and period are grouped. `age_interval` and `period_interval` indicate the width of age and period intervals, respectively. `age_group` and `period_group` are character vectors listing possible age and period groups. There are two ways to define age and period groups with unequal intervals: 1) defining `age_interval` and `period_interval`, or 2) defining `age_group` and `period_group`. Users must define age and period groups using one of the two options when `unequal_interval` is TRUE.
- `...`: further optional arguments to be passed to the model.

As mentioned in section [Coding scheme and contrast](#), we use effect coding to estimate the APC-I model. The `age` and `period` arguments in function `apci` accept categorical variables. Different from the common approach of dummy coding or simple coding, where an effect is defined as the difference of each group from the reference group, function `apci` uses effect coding (i.e., the sum-to-zero coding, deviation coding, or the ANOVA coding) as the default coding. Under this coding scheme, the effect of the omitted category equals the negative sum of the effects of all other categories. Computation wise, the effect coefficient of the omitted category is redundant because of the coding scheme. However, for the purpose of quantifying cohort effects as deviations from the main effects of age and period, it is useful to compute estimates for all age-by-period cross-classifications and their standard errors. For data with A age groups and P periods, therefore, function `apci` returns A and P number of main effect estimates and $A * P$ number of interaction estimates along with their standard error estimates. For the

⁵The following error may appear: “Error in solve.default(V): ‘a’ is 0-dim!”. To address this error, users may bypass the F test by setting the argument `dev.test` to FALSE.

main effects of age or period, the estimate can be interpreted as the deviation associated with each age or period group from the global mean. The age-by-period interactions represent the deviation from the expectation determined by the corresponding age and period main effects.

Also note that when age and period groups have unequal interval widths in an age-period classification table, the age-by-period interactions contained in a cohort no longer lie on the same diagonals. Because cohort effects are conceptualized and estimated as a structure of the age-by-period interactions in the APC-I model, it is technically possible to use the argument `unequal_interval` in package **APCI** to extract interaction coefficient estimates that lie on different diagonals. However, unequal age and period group intervals may complicate the issue of cohort overlapping noted by [Kupper et al. \(1985\)](#). For this reason, we do not recommend using unequal interval widths for age and period groups if possible.

After fitting the APC-I model, function `apci` will store the following components as a list for further usage:

- `model`: a summary of the fitted generalized linear regression model.⁶ It displays the standard regression output including coefficient and standard errors estimates.
- `dev_global`: the global F test results. It examines if the interaction terms are significant in a generalized linear regression model that contains age and period main effects and their interactions.
- `intercept`: the overall intercept (μ in equation 2).
- `age_effect`: a vector containing the estimated effect for each age group.
- `period_effect`: a vector containing the estimated effect for each time period.
- `cohort_average`: a vector containing the inter-cohort average deviations for comparing differences between cohorts.
- `cohort_slope`: a vector containing intra-cohort life-course trends.

Function tests

As mentioned earlier, the first step of implementing the APC-I model is to conduct a global F test of the age-by-period interactions. This step is a routine in function `apci`, but the procedure does not stop even if there is no statistically significant deviation from the age main effect and period main effect. Therefore, we recommend separately conducting the global F test. In R package **APCI**, the function tests can be used for this purpose. It can be used as follows:

```
tests(model, A, P, C, ...)
```

and takes the following arguments:

- `model`: a generalized linear regression model generated from the internal function `temp_model`.⁷
- `A, P, C`: numbers of age groups, period groups, or cohort groups. If age and period groups are of different widths, the values of will be automatically generated by the function.

Function `tests` will return a standard F test result including the value of the F test statistic and the associated p-value.

Functions for visualization

In package **APCI**, we provide four functions to facilitate visualizing the data and model results, namely `apci.plot.raw`, `apci.heatmap`, `apci.plot.hexagram`, and `apci.plot`, in different stages of a research project. They take similar input arguments. A summary of these arguments is given below.

Function `apci.plot.raw` is designed to plot the outcome variable aggregated by age or period groups. This function may be used in the stage of data exploration. Functions `apci.heatmap` and `apci.plot.hexagram` are designed to plot the age-by-period interactions from the APC-I model. Both functions generate heatmaps, where one axis represents age groups, and the other period groups. The cells in a diagonal represent one cohort. The difference between the two functions is the layout of the heatmap; one is a rectangular graph and the other a hexagram. Function `apci.plot` can be used to visualize both raw data and model results. It divides the canvas into four (2×2) panels. Three

⁶**APCI** supports all types of generalized linear regression models.

⁷`temp_model` is an internal function in R package **APCI** that accepts the same input arguments as those in function `apci`. Detailed syntax of this function can be found in [APCI reference manual](#).

of the four panels can be used to visualize the three effect estimates in the APC-I model and the left panel to add notes. For data exploration, users can visualize the mean values of the outcome variable across age, period, and cohort groups on the same canvas. Users can use the same functions to visualize the estimated age, period, and cohort effects from the modeling results.

The visualization functions in package `APCI` include:

```
apci.plot.raw(data, outcome_var, age, period, ...)

apci.plot.heatmap(model, age, period, color_map = NULL, color_scale = NULL,
                  quantile = NULL, ...)

apci.plot.hexagram(model, age, period, first_age, first_period, interval,
                  color_scale = NULL, color_map = NULL, quantile = NULL, ...)

apci.plot(model, age, period, outcome_var, type = "model", quantile = NULL,
          ...)
```

and takes the following arguments:

- `model`: a list recording the results from function `apci`.
- `outcome_var`: an object of class `character` indicating the name of the outcome variable used in the model. The outcome variable can be a continuous, binary, categorical, or count variable.
- `age`: a vector indicating the age group. The vector should be converted to a factor (or the terms of “category” and “enumerated type”).
- `period`: a vector indicating the time period. The vector should be converted to a factor (or “category”, “enumerated type”).
- `color_map`: a vector representing a color palette to be used in the figure. The default setting is greys if `color_map` is `NULL`. Alternatives, for example, can be `c("blue", "yellow")`, “blues”, etc.
- `cohort_scale`: a vector containing two values to indicate the minimum and maximum values, respectively, of the estimated cohort effects to be displayed. If `NULL`, the function will use the range from the estimation results.
- `quantile`: a number valued between 0 and 1, representing the desirable percentiles to be used in visualizing the data or model. If `NULL`, the original scale of the outcome variable will be used.

4 Examples: Application of the R package `APCI` to empirical data

We now illustrate how to use package `APCI`'s visualization and analytical functions. We describe and analyze two empirical datasets to demonstrate how this package may be used to analyze pooled cross-sectional data. We later briefly describe how to fit an APC-I model for multi-cohort longitudinal data.

Cross-sectional data of labor force participation in the United States

Temporal trends in men's and women's labor force participation (LFP) in United States have gathered much scholarly attention (see e.g. [Connelly, 1992](#); [Farkas, 1977](#); [Hollister and Smith, 2014](#); [Macunovich, 2012](#); [Treas, 1987](#); [Wilkie, 1991](#)). For example, whereas men's LFP steadily declined in the past decades ([Wilkie, 1991](#)), women's LFP continued to rise until the 1990s and the 2000s. Female LFP has since then reached a plateau and even begun to decline. Researchers have debated about the causes of this leveling off or decline. Some studies attributed the observed trends to period-specific factors such as labor demand ([Erceg and Levin, 2014](#)), the economic shocks of the Great Recession ([Boushey, 2005](#); [Hoffman, 2009](#)), social welfare and disability insurance ([Duggan and Imberman, 2009](#)), and gender role attitudes ([Fortin, 2015](#)).

However, the temporal trends in LFP are unlikely due to a pure period process. For example, because individuals may begin to leave the labor force in age 50, a decline in LFP should be expected if the proportion of the population age 50 or older has increased. That is, the recent trends may reflect a change in the age composition of the US population ([Aaronson et al., 2014](#)). The cohort succession may also contribute to the trend, a process in which older cohorts with higher or lower LFP rates begin to decrease and younger cohorts with lower or higher LFP enter the labor force [Lee \(2014\)](#). At the same time, critical social and demographic changes in education level, fertility, and attitudes about women working outside the home may be more of cohort-specific than period-specific processes because these

forces only affect individuals of certain ages (Balleer et al., 2014; Farré and Vella, 2013; Fernández, 2013; Goldin, 2006).

Given that the observed trends in LFP are likely a mix of age, period, and/or cohort patterns, an APC analysis that decomposes the temporal trends into age-, period-, and cohort-related variations is thus helpful for revealing the demographic, social, and economic changes that have underlined the temporal trends in American's labor force participation. In this following section, we demonstrate how to use the functions in **APCI** to undertake an APC analysis of men's and women's LFP using a cross-sectional dataset.

The Current Population Survey (CPS, Flood et al., 2021) is the primary source of labor force statistics in the United States (Flood et al., 2021). Beginning in the 1960s, the CPS has been collecting data on key demographic, economic, and education topics. We subset the 1990-2019 CPS data by gender, resulting in two datasets, namely *cpsmen* and *cpswomen*, to show how to conduct an APC-I analysis of men's and women's LFP in the United States using package **APCI**.

Datasets *cpsmen* and *cpswomen* contain a subset of men and women age 20-64 who participated in the 1990 to 2019 CPS. The following code is used to load the data into the working environment:

```
> data(cpsmen)
> data(cpswomen)
```

The first five rows of the datasets of *cpsmen* and *cpswomen* are:

```
> head(cpsmen, n = 5)
  asewt year age labforce educ
2854.84   3   5         0     1
1576.54   4   4         1     2
2340.55   2   6         1     3
 158.44   3   5         0     0
 347.09   6   6         1     3

> head(cpswomen, n = 5)
  asewt year age labforce educ
2415.67   2   3         1     1
 663.89   2   5         1     3
1653.01   6   4         1     2
1613.31   6   4         0     2
 177.23   4   3         1     2
```

where *labforce* indicates the respondent's labor force participation status (1=in the labor force, 0=not in the labor force). *asewt* is the person-level weight that the CPS recommends to be used in individual-level data analyses. *year* indicates the survey year when respondent was interviewed, grouped into 6 period groups (1=1990-94, 2=1995-99, ..., 6=2015-19). *age* indicates the respondent's age categories (1=20-24, 2=25-29, ..., 9=60-64). *educ* is a three-level categorical education measure (1=less than high school, 2=high school graduate, 3=college degree or above).

For data exploratory purpose, function `apci.plot.raw` visualizes the outcome variable in the following way:

```
> apci.plot.raw(data, outcome_var, age, period)
```

Figure 1 shows LFP rates by age groups (top panel) and period groups (bottom panel), respectively, for male (left panel) and female CPS respondents (right panel) age 20 to 64 from 1990 to 2019. Figure 1's top panel suggests similar age patterns in LFP across time periods. The bottom panel shows distinct period trends depending on age groups. For example, the LFP rates among women in the 55-59 and 60-64 age groups seem to have gone up whereas other age groups show a relatively flat trend. Such distinct period patterns in LFP by age groups suggest potential cohort variations in women's labor force participation. For men's LFP, however, the visualization results suggest that a simpler model with age and period main effects may suffice for summarizing their LFP patterns.

Function `apci` can be used to fit an APC-I model for pooled cross-sectional data or multi-cohort longitudinal/panel data. In the simplest form of an APC-I model without covariates for pooled cross-sectional data, function `apci` is called as follows:

```
> no_cov <- APCI::apci(outcome = "labforce",
+                      age = "age",
+                      period = "year",
+                      weight = "asewt",
+                      data = cpswomen,
```

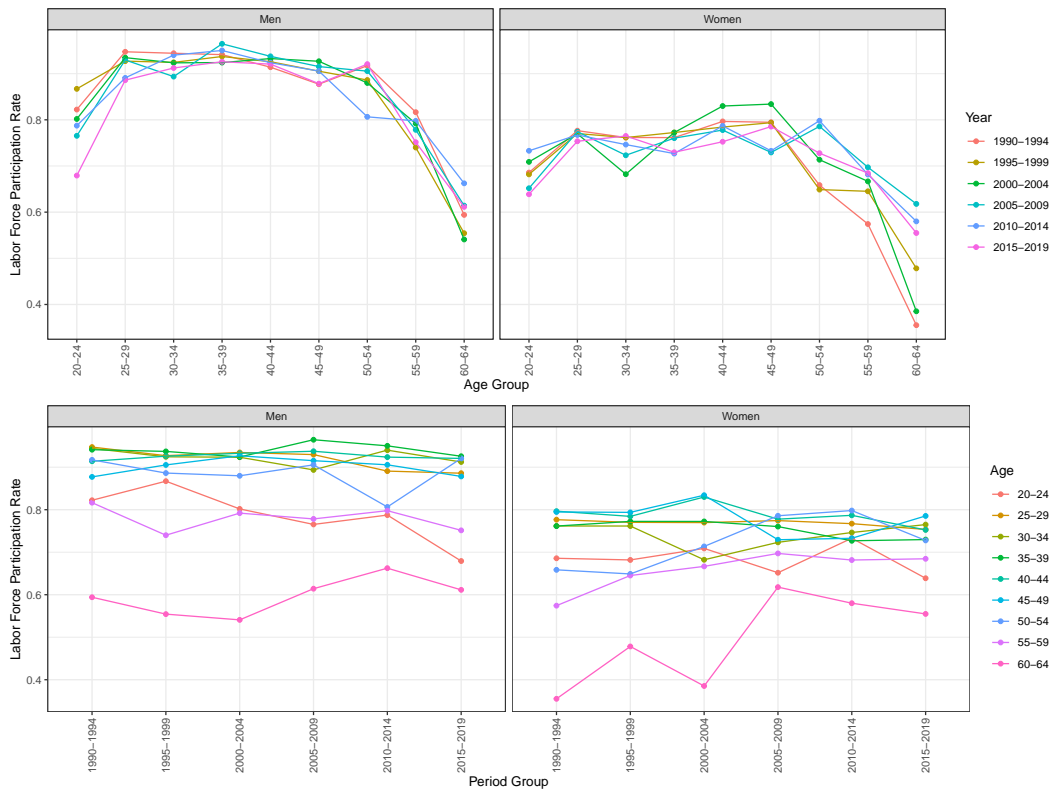


Figure 1: Period-specific age trajectories and age-specific period trends in labor force participation rates for men and women in the United States, the Current Population Survey 1990-2019. Age trajectories are similar across time periods for both men and women (top panel). Period trends differ by age groups, especially among women (bottom panel).

```
+ dev.test = FALSE,
+ family = "binomial")
```

It is often desirable to add covariates in the model, which can be done by calling the covariate argument. For example, suppose one would like to add education levels (“educ”) as a covariate in the model, function apci can be used as:

```
> with_cov <- apci::apci(outcome = "labforce",
+ age = "age",
+ period = "year",
+ covariate = c("educ"),
+ weight = "asecwt",
+ data = cpswomen,
+ print=F,
+ dev.test=FALSE,
+ family = "binomial")
```

Below is a summary of the results from an APC-I model that includes education levels as a covariate:

```
> summary(with_cov)
      Length Class      Mode
model      33  svyglm      list
dev_global  0  -none-      NULL
intercept  4  -none-      character
age_effect 45  -none-      character
period_effect 30 -none-      character
cohort_average 6 data.frame list
cohort_slope 6 data.frame list
int_matrix  5 data.frame list
cohort_index 54 -none-      numeric
data       7 data.frame list
```

The returned value is a list of objects. `model` contains the results from a logistic regression model with age and period main effects and the unstructured interactions. `dev_global` displays the global F test result. A significant F statistic suggests that there may exist cohort effects. `intercept` is the overall intercept (μ in Equation 2). `age_effect` gives estimated age main effect. `period_effect` is the estimated period main effect. `cohort_average` gives inter-cohort average deviations from age and period main effects. `cohort_slope` gives intra-cohort life-course linear slopes, which can be used for testing intra-cohort life-course dynamics. `int_matrix` displays a matrix that contains the estimated coefficients for age-by-period interactions. Note that there are A*P interactions in `int_matrix` because effect coding is used to compute the A+P-1 interaction estimates based on the (A-1)*(P-1) freely varying interaction parameters. Such interaction estimates are used to generate heatmaps similar to Figure 2. `data` stores the data fed into `apci` function. Users may call an object to obtain detailed results. For example, by calling `with_cov$cohort_average` and `with_cov$cohort_slope`, users can obtain estimated inter-cohort average deviations and intra-cohort life-course slopes.

The output below shows education-adjusted inter-cohort average deviations in women's LFP from analyzing the *cpswomen* data using function `apci`.

```
> with_cov$cohort_average
  c_avg_group c_avg_est c_avg_se c_avg_t c_avg_p c_avg_sig
1           1   -0.329   0.193  -1.709   0.088
2           2   -0.155   0.142  -1.091   0.275
3           3   -0.162   0.114  -1.422   0.155
4           4    0.047   0.097   0.481   0.631
5           5    0.096   0.085   1.139   0.255
6           6    0.174   0.076   2.288   0.022      *
7           7    0.034   0.074   0.457   0.648
8           8   -0.036   0.074  -0.493   0.622
9           9    0.003   0.073   0.047   0.963
10          10   -0.072   0.081  -0.894   0.371
11          11    0.030   0.085   0.353   0.724
12          12   -0.029   0.102  -0.288   0.774
13          13   -0.080   0.131  -0.609   0.543
14          14   -0.103   0.170  -0.608   0.543
```

where `c_avg_group` indicates cohort membership (e.g., cohort 1=the 1930 birth cohort, cohort 2=the 1935 birth cohort,...,cohort 14=the 1995 birth cohort), `c_avg_est` is inter-cohort average deviation, `c_avg_se` is the standard error estimate for the average deviation, `c_avg_t` is the t test statistic for the average deviation, and `c_avg_p` and `c_avg_sig` are the p values and alpha levels (*: $p < .05$, **: $p < .01$, and ***: $p < .001$), respectively.

The results from `with_cov$cohort_average` imply that on average, the LFP rates among cohort 6's—the 1955 birth cohort—significantly differ from the expected rates based on age and period main effects. Specifically, the 1955 cohort shows a .19 ($\exp(.174)-1$, $p < .05$) higher participation rate than the expectation based on the age and period main effects.

The output below shows education-adjusted intra-cohort life-course dynamics in women's LFP from analyzing the *cpswomen* data using function `apci`.

```
> with_cov$cohort_slope
  c_slp_group c_slp_est c_slp_se c_slp_t c_slp_p c_slp_sig
1           1         NA         NA         NA         NA      <NA>
2           2    0.165    0.195    0.849    0.396
3           3   -0.215    0.187   -1.148    0.251
4           4    0.163    0.189    0.866    0.386
5           5    0.093    0.184    0.508    0.611
6           6    0.007    0.169    0.039    0.969
7           7    0.047    0.172    0.277    0.782
8           8   -0.096    0.181   -0.530    0.596
9           9   -0.187    0.173   -1.076    0.282
10          10   -0.106    0.176   -0.602    0.547
11          11   -0.279    0.159   -1.750    0.080
12          12    0.353    0.160    2.207    0.027      *
13          13   -0.047    0.180   -0.262    0.793
14          14         NA         NA         NA         NA      <NA>
```

where `c_slp_group` indicates cohort membership (e.g., cohort 1=the 1930 birth cohort, cohort 2=the 1935 birth cohort,..., cohort 14=the 1995 birth cohort), `c_slp_est` is intra-cohort life-course slopes,

`c_slp_se` is the standard error estimate for the life-course slope, `c_slp_t` is the t test statistic for the life-course slope, and `c_slp_p` and `c_slp_sig` are the p values and alpha levels (*: $p < .05$, **: $p < .01$, and ***: $p < .001$), respectively. NAs are generated for the youngest and oldest cohort because there is only one age-by-period combination observed for the two cohorts and thus intra-cohort life-course dynamics cannot be accessed.

For example, for cohort 12 (the 1985 birth cohort), the estimated intra-cohort slope is 0.353 ($p < .05$), meaning that this cohort's LFP is lower than expected when they were young but higher than expected in older ages. Interestingly, for cohort 12 (the 1985 birth cohort), their average cohort deviation is not statistically significant. Such an insignificant inter-cohort average deviation and a significantly negative intra-cohort slope indicate a compensation life-course pattern; that is, this cohort's lower-than-expected LFP in younger ages seems to be compensated by their higher LFP when they were older.

The intra-cohort life-course dynamics are based on the age-by-period interactions as follows:

```
# the first six rows of the life-course dynamics
> with_cov$int_matrix
  iaesti iase iap iasig cohortindex
1  0.166 0.169 0.327                9
2 -0.048 0.207 0.818                8
3  0.068 0.164 0.678                7
4  0.095 0.172 0.581                6
5  0.205 0.191 0.283                5
6  0.227 0.193 0.239                4
# [there are 48 rows compressed]
```

where "iaesti" is the age-by-period interaction estimates, "iase" is the standard error estimate for the interaction term, "iap" and "iasig" are the p value and alpha level (*: $p < .05$, **: $p < .01$, and ***: $p < .001$), respectively, and "cohortindex" indicates cohort membership.

The following code can be used to organize the intra-cohort life-course estimates in a matrix form:

```
> matrix(with_cov$int_matrix, A, P)[A:1,]
# A is the number of age groups and P is the number of period groups.
```

	period #1	period #2	period #3	period #4	period #5	period #6
age #9	-0.329	-0.038	-0.416*	0.334	0.267	0.182
age #8	-0.272	0.043	0.017	0.125	0.001	0.086
age #7	-0.112	-0.392*	-0.070	0.257	0.278	0.040
age #6	0.227	-0.046	0.443*	-0.333*	-0.258	-0.033
age #5	0.205	0.066	0.061	-0.150	-0.074	-0.107
age #4	0.095	0.044	0.139	0.065	-0.109	-0.234
age #3	0.068	0.059	-0.359*	-0.147	0.096	0.283
age #2	-0.048	0.256	-0.006	0.067	-0.156	-0.113
age #1	0.166	0.009	0.191	-0.216	-0.046	-0.103

Based on the R package `ggplot2` (Wickham, 2016), heatmaps can be generated to visualize inter- and intra-cohort patterns and motivate a subsequent formal APC analysis. For example, for dataset *whitemen*, both inter-cohort average deviations and intra-cohort life-course dynamics may be visualized in a heatmap as follows:

```
> apci.plot.heatmap(model = with_cov, age = "age", period = 'year',
  color_map = c('blue', 'yellow'))
```

Figure 2 is a heatmap of the estimated age-by-period interactions, with rows defined by age groups and columns by time periods. Each square represents an age-by-period interaction. Yellow squares indicate lower participation rates than the expectation determined by the age and period main effects. Blue squares indicate higher-than-expected rates. Each diagonal that runs from the lower left to the upper right represents one birth cohort. The dotted line in Figure 2 indicates a significant inter-cohort average deviation but an insignificant intra-cohort slope in their LFP for the 1955 cohort. Cohorts that on average significantly deviate from the expected rates based on age and period main effects are indicated by solid, dashed, or dotted lines. Solid lines indicate significantly ($p < .05$) positive intra-cohort life-course slopes, dashed lines significantly negative intra-cohort slopes, and dotted lines significant average inter-cohort deviations but insignificant intra-cohort slopes. Users have the options to customize the elements of these figures to suit their research or teaching purposes.

Figure 2 indicates that although some cohorts had LFP rates that differ from the expected rates based on age and period main effects, only the 1955 birth cohort (marked by a dotted line) had, on

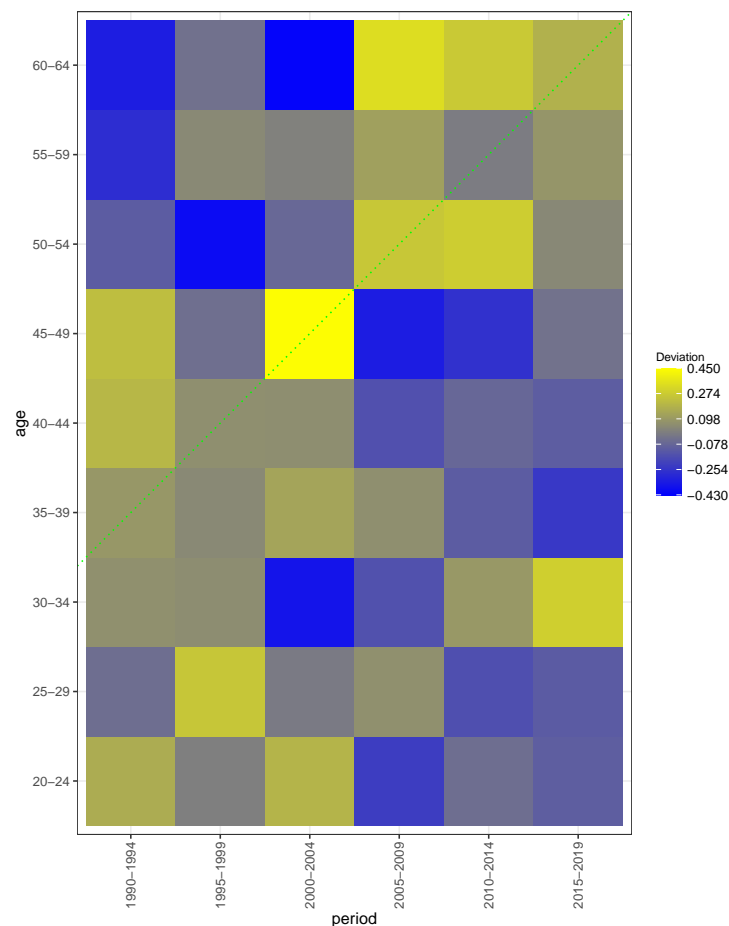


Figure 2: Cohort deviation heatmap showing higher-than-expected labor force participation rates that persist over their life course for the 1955 birth cohort. Each square represents an age-by-period interaction. Yellow squares indicate lower participation rates than the expectation determined by the age and period main effects. Blue squares indicate higher-than-expected rates. Each diagonal that runs from the lower left to the upper right represents one birth cohort. The dotted line indicates a significant average inter-cohort deviation but an insignificant intra-cohort slope in their labor force participation rates for the 1955 cohort.

average, higher-than-expected LFP rates (inter-cohort average deviation = .174, $p < .05$), and this higher LFP seems to persist over their life course (intra-cohort slope = .007, $p = .969$). This visualization results are consistent with the results from `with_cov$cohort_average` and `with_cov$cohort_slope`.

Users can also use bar plots to visualize inter-cohort average deviations. Function `apci.bar` can be used as:

```
> apci.bar(model = with_cov, age = "age", period = "year",
           cohort_label = seq(1930, 1995, 5))
```

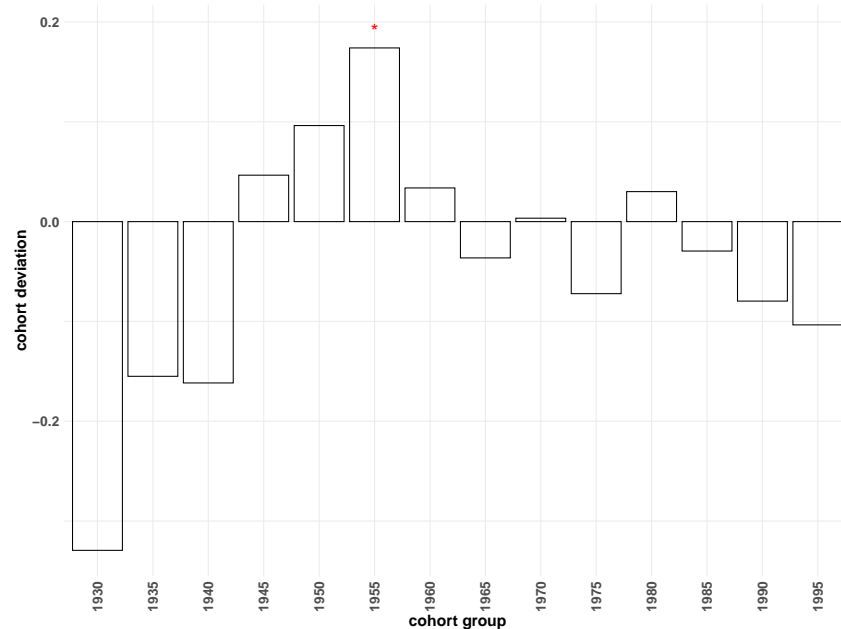


Figure 3: Bar plots showing inter-cohort average deviations in women's labor force participation rates. Only the 1955 birth cohort had a significantly higher-than-expected participation rate. The horizontal line at $y=0$ indicates expected labor force participation rates for each cohort based on the main effects of age and period when they were observed. The bars indicate the estimated average deviation for each cohort from the age and period main effects. Bars above the horizontal line indicate positive average deviations, and bars below the line indicate negative average deviations. The asterisk sign indicates that a cohort's average deviation is significantly different from the expectation determined by age and period main effects at the .05 or lower level.

Figure 3 illustrates inter-cohort deviations based on the average of the age-by-period interaction estimates contained in each cohort. The horizontal line at $y=0$ indicates expectations for all cohort based on the main effects of age and period when they are observed. The bars indicate the estimated average deviation for each cohort from the age and period main effects. Bars above the horizontal line indicate positive average deviations, and bars below the line indicate negative average deviations. The asterisk sign indicates that a cohort's average deviation is significantly different from the expectation determined by age and period main effects. Figure 3 suggests some cohort variation in women's LFP using the 1990-2019 CPS data, but only the 1955 birth cohort had significantly higher-than-expected LFP rates.

Longitudinal data

For longitudinal data (i.e., panel data, repeated measure data) that include multiple cohorts, users may set the argument `gee` to `TRUE` to estimate an APC-I model using the generalized estimating equation (GEE) technique (Liang and Zeger, 1986; Carey et al., 2019). When `gee` is `TRUE`, users will also need to specify arguments `id` and `corstr` accordingly.

```
> model_gee <- apci(outcome = "y",
+                   age = "age",
+                   period = "period",
+                   cohort = NULL,
+                   weight = NULL,
+                   covariate = NULL,
```



```

+         data=simulation_gee,
+         family = "gaussian",
+         dev.test = FALSE,
+         print = TRUE,
+         gee = TRUE,
+         id = "id",
+         corstr = "exchangeable")
> summary(model_gee)

```

The list of output results is similar to that for pooled cross-sectional data, but the standard errors are corrected using the GEE's sandwich estimator.

5 Use the R package APCI in Stata

We also designed a Stata command `apci` based on the Stata command `rcall` (Haghish, 2019) to help implement APC-I models in Stata. The command is used as:

```
apci depvar [indepvars], outcome(depvar) age(age) period(period) family("gaussian") [if] [in] [weight]
```

Stata users can use the above command to fit APC-I models and obtain all the results as in R. A Stata ado file for installing this command can be downloaded at <https://sites.psu.edu/liyingluo/software/>.

6 Conclusion and future development

In this article, we introduced an R package `APCI` and Stata command `apci` for implementing the age-period-cohort-interaction (APC-I) model developed by Luo and Hodges (2020a). In addition to pooled cross-sectional data analysis, we extended the package to permit multi-cohort longitudinal or panel data analysis. This package also contains a set of visualization tools to help researchers motivate an APC analysis and interpret the results. We clarify the implications of coding schemes for estimating and testing main effects and interaction effects in the APC-I model. We illustrate how to use this package using the empirical examples of labor force participation using the 1990-2019 data from the Current Population Survey.

Luo and Hodges (2020a) described a local F test for testing the variation associated with the multiple age-by-period interactions contained in each cohort. Because the parameterization of the local F test is more intricate than it appears, the R package `APCI` and the Stata command `apci` currently do not support such tests as of version 1.0.5 but may be available in later versions.

Moreover, it may be of interest to examine the interaction effects of cohort and other explanatory variables such as education and geographic areas. Because cohort effects are conceptualized and operationalized as a two-way interaction term of age and period effects, an interaction term between cohort and another variable is equivalent to a three-way interaction among age, period, and another variable. Future development may consider creating functions to facilitate summarizing and interpreting the more complex three-way interaction terms in the APC-I model.

Bibliography

- S. Aaronson, T. Cajner, B. Fallick, F. Galbis-Reig, C. Smith, and W. Wascher. Labor Force Participation: Recent Developments and Future Prospects. *Brookings Papers on Economic Activity* 2014, (2): 197–275, 2014. URL <https://www.brookings.edu/bpea-articles/labor-force-participation-recent-developments-and-future-prospects/>. [p84]
- L. S. Aiken, S. G. West, and R. R. Reno. *Multiple Regression: Testing and Interpreting Interactions*. SAGE, 1991. ISBN 978-0-7619-0712-1. Google-Books-ID: LcWLUyXcmnkC. [p80]
- D. F. Alwin and R. J. McCammon. Generations, Cohorts, and Social Change. In J. T. Mortimer and M. J. Shanahan, editors, *Handbook of the Life Course*, Handbooks of Sociology and Social Research, pages 23–49. Springer US, Boston, MA, 2003. ISBN 978-0-306-48247-2. doi: 10.1007/978-0-306-48247-2_2. URL https://doi.org/10.1007/978-0-306-48247-2_2. [p77]
- A. Balleer, R. Gomez-Salvador, and J. Turunen. Labour force participation across Europe: A cohort-based analysis. 46(4):1385–1415, 2014. ISSN 1435-8921. URL https://econpapers.repec.org/article/sprempeco/v_3a46_3ay_3a2014_3ai_3a4_3ap_3a1385-1415.htm. [p85]

- H. Boushey. Are Women Opting Out? Debunking the Myth. *Center for Economic and Policy Research (CEPR)*, pages 2005–36, 2005. [p84]
- V. J. Carey, T. S. Lumley, C. Moler, and B. Ripley. *gee: Generalized Estimation Equation Solver*, 2019. URL <https://CRAN.R-project.org/package=gee>. R package version 4.13-20. [p90]
- L. Chauvel, A. K. Leist, and V. Ponomarenko. Testing Persistence of Cohort Effects in the Epidemiology of Suicide: An Age-Period-Cohort Hysteresis Model. *PLOS ONE*, 11(7):e0158538, July 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0158538. [p79]
- C. C. Clogg. Cohort Analysis of Recent Trends in Labor Force Participation. *Demography*, 19(4):459–479, Nov. 1982. ISSN 0070-3370, 1533-7790. doi: 10.2307/2061013. URL <https://link.springer.com/article/10.2307/2061013>. [p77]
- R. Connelly. The Effect of Child Care Costs on Married Women’s Labor Force Participation. *The Review of Economics and Statistics*, 74(1):83–90, 1992. [p84]
- D. Dannefer. Aging as Intracohort Differentiation: Accentuation, the Matthew Effect, and the Life Course. *Sociological Forum*, 2(2):211–236, 1987. ISSN 0884-8971. URL <https://www.jstor.org/stable/684472>. Publisher: [Wiley, Springer]. [p79]
- M. Duggan and S. A. Imberman. Why are the disability rolls skyrocketing? the contribution of population characteristics, economic conditions, and program generosity. In *Health at Older Ages*, pages 337–380. University of Chicago Press, 2009. [p84]
- C. J. Erceg and A. T. Levin. Labor Force Participation and Monetary Policy in the Wake of the Great Recession. *Journal of Money, Credit and Banking*, 46(S2):3–49, 2014. ISSN 1538-4616. doi: 10.1111/jmcb.12151. [p84]
- G. Farkas. Cohort, age, and period effects upon the employment of white females: Evidence for 1957–1968. *Demography*, 14(1):33–42, 1977. [p84]
- L. Farré and F. Vella. The Intergenerational Transmission of Gender Role Attitudes and its Implications for Female Labour Force Participation. *Economica*, 80(318):219–247, 2013. ISSN 1468-0335. doi: 10.1111/ecca.12008. [p85]
- R. Fernández. Cultural Change as Learning: The Evolution of Female Labor Force Participation over a Century. *American Economic Review*, 103(1):472–500, 2013. ISSN 0002-8282. doi: 10.1257/aer.103.1.472. URL <https://www.aeaweb.org/articles?id=10.1257/aer.103.1.472>. [p85]
- K. F. Ferraro and P. M. Morton. What Do We Mean by Accumulation? Advancing Conceptual Precision for a Core Idea in Gerontology. *The Journals of Gerontology: Series B*, 73(2):269–278, Jan. 2018. ISSN 1079-5014. doi: 10.1093/geronb/gbv094. URL <https://academic.oup.com/psychsocgerontology/article/73/2/269/2632120>. [p79]
- S. E. Fienberg and W. M. Mason. Identification and Estimation of Age-Period-Cohort Models in the Analysis of Discrete Archival Data. *Sociological Methodology*, 10:1–67, 1979. ISSN 0081-1750. doi: 10.2307/270764. URL <http://www.jstor.org/stable/270764>. [p78]
- S. E. Fienberg and W. M. Mason. Specification and Implementation of Age, Period and Cohort Models. In *Cohort Analysis in Social Research*, pages 45–88. Springer, New York, NY, 1985. ISBN 978-1-4613-8538-7 978-1-4613-8536-3. doi: 10.1007/978-1-4613-8536-3_3. URL https://link.springer.com/chapter/10.1007/978-1-4613-8536-3_3. [p78]
- S. Flood, M. King, R. Rodgers, S. Ruggles, R. J. Warren, and M. Westberry. Integrated public use microdata series, current population survey: Version 9.0 [dataset]. *Minneapolis, MN: IPUMS*, 2021. doi: <https://doi.org/10.18128/D030.V9.0>. [p77, 85]
- N. M. Fortin. Gender Role Attitudes and Women’s Labor Market Participation: Opting-Out, AIDS, and the Persistent Appeal of Housewifery. *Annals of Economics and Statistics*, (117/118):379–401, 2015. ISSN 2115-4430. doi: 10.15609/annaeconstat2009.117-118.379. [p84]
- E. Fosse and C. Winship. Analyzing Age-Period-Cohort Data: A Review and Critique. *Annual Review of Sociology*, 45(1):467–492, 2019. doi: 10.1146/annurev-soc-073018-022616. URL <https://doi.org/10.1146/annurev-soc-073018-022616>. [p77, 78]
- C. Goldin. The Quiet Revolution That Transformed Women’s Employment, Education, and Family. *The American Economic Review*, 96(2):1–21, 2006. ISSN 0002-8282. URL <http://www.jstor.org/stable/30034606>. [p85]

- E. F. Haghish. Seamless interactive language interfacing between R and Stata. *The Stata Journal*, 19 (1):61–82, Mar. 2019. ISSN 1536-867X. doi: 10.1177/1536867X19830891. URL <https://doi.org/10.1177/1536867X19830891>. Publisher: SAGE Publications. [p91]
- J. Heckman and R. Robb. Using Longitudinal Data to Estimate Age, Period and Cohort Effects in Earnings Equations. In W. M. Mason and S. E. Fienberg, editors, *Cohort Analysis in Social Research*, pages 137–150. Springer New York, 1985. ISBN 978-1-4613-8538-7 978-1-4613-8536-3. doi: 10.1007/978-1-4613-8536-3_5. URL http://link.springer.com/chapter/10.1007/978-1-4613-8536-3_5. [p77]
- J. Hobcraft, J. Menken, and S. Preston. Age, Period, and Cohort Effects in Demography: A Review. *Population Index*, 48(1):4–43, 1982. ISSN 0032-4701. doi: 10.2307/2736356. URL <http://www.jstor.org/stable/2736356>. [p77]
- S. D. Hoffman. The changing impact of marriage and children on women’s labor force participation. *Monthly Lab. Rev.*, 132:3, 2009. [p84]
- M. N. Hollister and K. E. Smith. Unmasking the Conflicting Trends in Job Tenure by Gender in the United States, 1983–2008. *American Sociological Review*, 79(1):159–181, Feb. 2014. ISSN 0003-1224. doi: 10.1177/0003122413514584. [p84]
- J. Jaccard and R. Turrisi. *Interaction Effects in Multiple Regression*. SAGE, Mar. 2003. ISBN 978-0-7619-2742-6. Google-Books-ID: n0pIZTQqvmIC. [p80]
- L. L. Kupper, J. M. Janis, I. A. Salama, C. N. Yoshizawa, B. G. Greenberg, and H. H. Winsborough. Age-Period-Cohort Analysis: An Illustration of the Problems in Assessing Interaction in One Observation Per Cell Data. *Communications in Statistics - Theory and Methods*, 12(23):201–217, Jan. 1983. ISSN 0361-0926. doi: 10.1080/03610928308828640. URL <https://doi.org/10.1080/03610928308828640>. [p78]
- L. L. Kupper, J. M. Janis, A. Karmous, and B. G. Greenberg. Statistical Age-Period-Cohort Analysis: A Review and Critique. *Journal of Chronic Diseases*, 38(10):811–830, Jan. 1985. ISSN 0021-9681. doi: 10.1016/0021-9681(85)90105-5. URL <http://www.sciencedirect.com/science/article/pii/0021968185901055>. [p78, 83]
- J. Y. Lee. The Plateau in U.S. Women’s Labor Force Participation: A Cohort Analysis. *Industrial Relations: A Journal of Economy and Society*, 53(1):46–71, Jan. 2014. ISSN 1468-232X. doi: 10.1111/irel.12046. URL <http://onlinelibrary.wiley.com/doi/10.1111/irel.12046/abstract>. [p84]
- K.-Y. Liang and S. L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73 (1):13–22, 1986. [p90]
- Y. Lu and L. Luo. Cohort Variation in U.S. Violent Crime Patterns from 1960 to 2014: An Age-Period-Cohort-Interaction Approach. *Journal of Quantitative Criminology*, Oct. 2020. ISSN 1573-7799. doi: 10.1007/s10940-020-09477-3. URL <https://doi.org/10.1007/s10940-020-09477-3>. [p77]
- L. Luo. Paradigm shift in age-period-cohort analysis: A response to Yang and Land, O’Brien, Held and Riebler, and Fienberg. *Demography*, 50(6):1985–1988, 2013. [p78]
- L. Luo and J. S. Hodges. The Age-Period-Cohort-Interaction Model for Describing and Investigating Inter-cohort Deviations and Intra-cohort Life-course Dynamics:. *Sociological Methods & Research*, Jan. 2020a. doi: 10.1177/0049124119882451. URL <https://journals.sagepub.com/doi/10.1177/0049124119882451>. Publisher: SAGE PublicationsSage CA: Los Angeles, CA. [p77, 78, 79, 81, 91]
- L. Luo and J. S. Hodges. Constraints in Random Effects Age-Period-Cohort Models. *Sociological Methodology*, 50(1):276–317, Aug. 2020b. ISSN 0081-1750. doi: 10.1177/0081175020903348. URL <https://doi.org/10.1177/0081175020903348>. Publisher: SAGE Publications Inc. [p78]
- L. Luo, J. Hodges, C. Winship, and D. Powers. The Sensitivity of the Intrinsic Estimator to Coding Schemes: Comment on Yang, Schulhofer-Wohl, Fu, and Land. *American Journal of Sociology*, 122(3): 930–961, Nov. 2016. ISSN 0002-9602. doi: 10.1086/689830. URL <http://www.journals.uchicago.edu/doi/abs/10.1086/689830>. [p78, 80]
- X. Ma. What are the temporal dynamics of taste? *Poetics*, page 101514, Dec. 2020. ISSN 0304-422X. doi: 10.1016/j.poetic.2020.101514. URL <http://www.sciencedirect.com/science/article/pii/S0304422X2030262X>. [p77]

- D. J. Macunovich. Relative Cohort Size, Relative Income, and Married Women's Labor Force Participation: United States, 1968–2010. *Population and Development Review*, 38(4):631–648, 2012. ISSN 1728-4457. doi: 10.1111/j.1728-4457.2012.00530.x. [p84]
- K. O. Mason, W. M. Mason, H. H. Winsborough, and W. K. Poole. Some Methodological Issues in Cohort Analysis of Archival Data. *American Sociological Review*, 38(2):242–258, 1973. ISSN 0003-1224. doi: 10.2307/2094398. URL <http://www.jstor.org/stable/2094398>. [p78]
- S. L. Morgan and J. Lee. A rolling panel model of cohort, period, and aging effects for the analysis of the general social survey. *Sociological Methods & Research*, page 00491241211043135, 2021. [p78]
- R. M. O'Brien. Estimable intra-age, intra-period, and intra-cohort effects in age-period-cohort multiple classification models. *Quality & Quantity: International Journal of Methodology*, 54(4):1109–1127, 2020. [p78, 79]
- B. A. Pescosolido, A. Halpern-Manners, L. Luo, and B. Perry. Trends in public stigma of mental illness in the US, 1996-2018. *JAMA network open*, 4(12):e2140202–e2140202, 2021. [p77]
- C. Robertson and P. Boyle. Age, period and cohort models: The use of individual records. *Statistics in Medicine*, 5(5):527–538, 1986. ISSN 1097-0258. doi: 10.1002/sim.4780050517. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.4780050517>. [p78]
- N. B. Ryder. The Cohort as a Concept in the Study of Social Change. *American Sociological Review*, 30(6):843–861, 1965. ISSN 0003-1224. doi: 10.2307/2090964. URL <http://www.jstor.org/stable/2090964>. [p78]
- S. Sarma, R. A. Devlin, A. Thind, and M.-K. Chu. Canadian family physicians' decision to collaborate: Age, period and cohort effects. *Social Science & Medicine*, 75(10):1811–1819, Nov. 2012. ISSN 0277-9536. doi: 10.1016/j.socscimed.2012.07.028. URL <http://www.sciencedirect.com/science/article/pii/S0277953612005692>. [p78]
- H. Scheffé. *The Analysis of Variance*. John Wiley & Sons, Mar. 1999. ISBN 978-0-471-34505-3. Google-Books-ID: h7NuoPIXh9UC. [p79]
- M. te Grotenhuis, B. Pelzer, L. Luo, and A. W. Schmidt-Catran. The Intrinsic Estimator, Alternative Estimates, and Predictions of Mortality Trends: A Comment on Masters, Hummer, Powers, Beck, Lin, and Finch. *Demography*, 53(4):1245–1252, Aug. 2016. ISSN 1533-7790. doi: 10.1007/s13524-016-0476-8. URL <https://doi.org/10.1007/s13524-016-0476-8>. [p78, 80]
- J. Treas. The Effect of Women's Labor Force Participation on the Distribution of Income in the United States. *Annual Review of Sociology*, 13:259–288, 1987. ISSN 0360-0572. [p84]
- A. M. Verdery, K. England, A. Chapman, L. Luo, K. McLean, and S. Monnat. Visualizing Age, Period, and Cohort Patterns of Substance Use in the U.S. Opioid Crisis. *Socius*, 6:2378023120906944, Jan. 2020. ISSN 2378-0231. doi: 10.1177/2378023120906944. URL <https://doi.org/10.1177/2378023120906944>. Publisher: SAGE Publications. [p77]
- H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL <https://ggplot2.tidyverse.org>. [p88]
- J. R. Wilkie. The Decline in Men's Labor Force Participation and Income and the Changing Structure of Family Economic Support. *Journal of Marriage and Family*, 53(1):111–122, 1991. ISSN 0022-2445. doi: 10.2307/353137. [p84]
- J. Xu and L. Luo. *APCI: A New Age-Period-Cohort Model for Describing and Investigating Inter-Cohort Differences and Life Course Dynamics*, 2021. URL <https://cran.r-project.org/web/packages/APCI/index.html>. R package version 1.0.5. [p77]
- Y. Yang and K. C. Land. A Mixed Models Approach to the Age-Period-Cohort Analysis of Repeated Cross-Section Surveys, with an Application to Data on Trends in Verbal Test Scores. *Sociological Methodology*, 36(1):75–97, Dec. 2006. ISSN 1467-9531. doi: 10.1111/j.1467-9531.2006.00175.x. URL <http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9531.2006.00175.x/abstract>. [p78]

Jiahui Xu
 The Pennsylvania State University
 917 Oswald Tower University Park, PA 16802
 United States

ORCID: 0000-0003-2728-0674
jpx5053@psu.edu

Liyang Luo
The Pennsylvania State University
202 Oswald Tower University Park, PA 16802
United States
(ORCID: 0000-0001-5393-6695)
liyangluo@psu.edu

shinybrms: Fitting Bayesian Regression Models Using a Graphical User Interface for the R Package brms

by Frank Weber, Katja Ickstadt, and Anne Glass

Abstract Despite their advantages, the application of Bayesian regression models is still the exception compared to frequentist regression models. Here, we present our R package `shinybrms` which provides a graphical user interface for fitting Bayesian regression models, with the frontend consisting of a `shiny` app and the backend relying on the R package `brms` which in turn relies on Stan. With `shinybrms`, we hope that Bayesian regression models (and regression models in general) will become more popular in applied research, data analyses, and teaching. Here, we illustrate our graphical user interface by the help of an example from medical research.

1 Introduction

The relevance of regression models in applied research has already been well pointed out, for example, by Karabatsos (2015):

“Regression modeling is ubiquitous in empirical areas of scientific research. This is because most research questions can be asked in terms of how a dependent variable changes as a function of one or more covariates (predictors).”

Conducting regression analyses in a *Bayesian* framework has a lot of advantages. Introductory texts on Bayesian statistics (in general) are, e.g., McElreath (2020), Albert and Hu (2019), Reich and Ghosh (2019), StataCorp (2019a), Gelman et al. (2020a), and Johnson et al. (2022). For readers with little background in Bayesian statistics, we recommend reading one of these textbooks first. A more detailed introduction may be found, e.g., in Gelman et al. (2014). In particular, Bayesian statistics has the following advantages (for further advantages, see, e.g., Gelman et al., 2014; StataCorp, 2019a):

1. Bayesian methods allow for incorporation of *prior knowledge*. Generally, inclusion of prior knowledge is desirable: The flat prior implied by the frequentist maximum likelihood (ML) method may lead to nonsensical inferences (Gelman et al., 2017). Even if the inclusion of informative prior knowledge is not desired, weakly informative priors have the advantage (compared to so-called “noninformative” priors¹) to downweight unreasonable parameter values and to introduce a certain *regularization* or *penalization*, helping against overfitting (Gelman, 2006; Gelman et al., 2008, 2014, 2017). Hereafter, we follow conventional notation in Bayesian statistics and denote the prior for the parameter vector θ by $p(\theta)$.
2. Similarly, the prior distribution may be used to impose *parameter constraints* in an easy and natural way. There is no need for *ad-hoc* solutions to cut off parameter estimates. For example, many frequentist between-study variance estimators in the random-effects meta-analysis model are cut off at zero.
3. It is usually possible to infer the posterior *exactly* (apart from minor approximations such as those arising from the Monte Carlo error). In that case, Bayesian statistics does not need to resort to large-sample approximations such as the asymptotic normal distribution of the ML estimator often used in frequentist statistics.
4. For most practical cases, Markov chain Monte Carlo (MCMC) sampling (see section [Markov chain Monte Carlo](#)) constitutes a *generic* Bayesian inference method. In frequentist statistics, generic methods such as the asymptotic normal distribution of the ML estimator can be unsatisfactory, e.g., for small sample sizes. This is why in frequentist statistics, different inferential methods have often evolved for the same task or model. This complicates frequentist analyses for users, especially for those with little background in statistics.
5. The quantities derived from the posterior have a *more intuitive interpretation* than their frequentist counterparts which are based on the sampling distribution of the estimator. In particular, Bayesian posterior intervals (credible intervals, CrIs) have the interpretation that is often incorrectly attributed to frequentist confidence intervals (CIs) (McElreath, 2020) and posterior tail-area probabilities have the interpretation that is often incorrectly attributed to frequentist p -values.

¹We added quotation marks here since noninformative priors might be more informative than intended (Gelman et al., 2017).

6. In Bayesian statistics, *uncertainty in nuisance parameters* is easily—and naturally—taken into account by integrating them out from the posterior:

$$p(\boldsymbol{\psi}|\mathcal{D}) = \int p(\boldsymbol{\psi}, \boldsymbol{\phi}|\mathcal{D}) \, d\boldsymbol{\phi} \quad (1)$$

with \mathcal{D} denoting the data, $\boldsymbol{\psi}$ the parameter vector of interest, and $\boldsymbol{\phi}$ the nuisance parameter vector (so that $(\boldsymbol{\psi}^\top, \boldsymbol{\phi}^\top)^\top = \boldsymbol{\theta}$)². Taking uncertainty in nuisance parameters into account helps against overfitting (like the penalization mentioned in enumeration point number 1 above), but in general, this is not that easy in frequentist statistics, as can be seen in random-effects meta-analyses (Weber et al., 2021).

7. When combined with probabilistic programming (as done, e.g., by the various sampling methods introduced in section [Algorithms for inferring the posterior](#)), a Bayesian analysis naturally *propagates the posterior uncertainty* into derived quantities (Gelman et al., 2020a).
8. Often, frequentist analyses result in the typical *null-hypothesis significance testing* which is being criticized to an increasing degree (Amrhein and Greenland, 2018; Amrhein et al., 2019; McShane et al., 2019). Null-hypothesis significance testing is especially problematic for null hypotheses consisting of only a point in parameter space. Of course, Bayesian analyses may also result in null-hypothesis significance testing or similar hypothesis-testing procedures, but in our experience, this is not as common as in frequentist analyses. We designed our software presented here in a way that does not encourage null-hypothesis significance testing.
9. *Posterior predictive checks (PPCs)*—which should be part of a Bayesian workflow (Gelman et al., 2020b)—are an easy and intuitive way of performing model diagnostics in a Bayesian framework, even though the choice and interpretation of the PPCs require some experience (Gelman et al., 2020b). In a frequentist framework, model diagnostics are often not that easy to perform, at least if the uncertainty from parameter estimation should be taken into account.

These advantages will be illustrated in the context of the example from section [Example](#), by comparing our Bayesian analysis to a frequentist one (see section “Frequentist analysis of the example” in the online Supplement file ‘Supplement_sections.pdf’).

Despite the aforementioned advantages, Bayesian methods—and Bayesian regression models (BRMs) in particular—are still not as common as their frequentist counterparts. In 2005, Woodward (2005) supposed one reason to be the lack of a “good user interface” which would allow applied researchers to fit BRMs as conveniently as other statistical methods for which a graphical user interface (GUI) already exists. In the meantime, several GUIs have emerged (see section [Existing GUIs](#)), but to our knowledge, until the first release of our R package **shinybrms** (Weber, 2022), there was no GUI which used Stan (Carpenter et al., 2017; Stan Development Team, 2022d) for inferring the posterior in BRMs. Stan has several advantages compared to other methods for inferring the posterior. In particular, it is highly flexible with respect to modeling choices and very efficient. Details will be given in section [Algorithms for inferring the posterior](#).

Our **shinybrms** package is noncommercial and available at the Comprehensive R Archive Network (CRAN). While **shinybrms**’s frontend is a **shiny** (Chang et al., 2022) app, **shinybrms**’s backend completely relies on **brms** (Bürkner, 2017, 2018) which itself relies on Stan. Both of **brms**’s backends (i.e., interfaces to Stan), namely **rstan** (Stan Development Team, 2022b) and **cmdstanr** (Gabry and Češnovar, 2022), are supported by **shinybrms**. For the inspection of the Stan output, the **shinystan** (Gabry, 2022) app may be launched from within **shinybrms**.

To explain the particular advantages of Stan in detail, we have to take a closer look at different ways for inferring the posterior. This is the purpose of section [Algorithms for inferring the posterior](#). In section [Existing GUIs](#), we summarize existing GUIs for BRMs. That section is partly influenced by Ramírez-Hassan and Graciano-Londoño (2021). In section [Features of shinybrms](#), we present the features of **shinybrms**. The usage of the **shinybrms** app is illustrated by the help of a real-world example in section [Example](#). Finally, we discuss our work in section [Discussion](#).

2 Algorithms for inferring the posterior

As mentioned above, in Bayesian statistics, uncertainty arising from the estimation of nuisance parameters is taken into account by integrating them out from the posterior. This is not the only integration occurring in posterior inference: Basically every quantity derived from the posterior is somehow connected to an integration over the posterior. However, it is the integration which also causes a lot of complications. While it is most desirable to perform posterior inference by exact

²Here, we are slightly abusing the notation by employing a single integral symbol for a possibly multiple integral.

calculation of the desired integrals (using analytic expressions), this approach is often infeasible and even if it is feasible, it has the downside of being not as flexible as other approaches since it needs to be tailored to the statistical model at hand. Numerical integration (e.g., by quadrature) may seem like a remedy, but is often only feasible up to a limited dimensionality of the parameter space. Depending on the algorithm, numerical integration may also introduce tuning quantities, hindering its “out-of-the-box” usage. Integration by simple Monte Carlo (MC) sampling may seem like an alternative, but this is only possible for distributions one may directly sample from (e.g., a Gaussian distribution).

Markov chain Monte Carlo

With the advent of Markov chain Monte Carlo (MCMC) methods, Bayesian inference has changed a lot (Woodward, 2005; Lunn et al., 2009). The first MCMC algorithm was the *Metropolis* algorithm (Metropolis et al., 1953) which starts from an initial point in parameter space and iteratively samples a *proposal*³ from a *symmetric*⁴ jumping distribution and accepts the proposal with a certain acceptance probability which depends on the ratio of the target (here, the posterior) density at the current position and at the proposal. The *Metropolis-Hastings (MH)* algorithm (Metropolis et al., 1953; Hastings, 1970) generalizes the Metropolis algorithm to asymmetric jumping distributions. *Gibbs sampling* (Geman and Geman, 1984; Gelfand and Smith, 1990) consists of alternately sampling from the full conditional posterior distributions and is a special MH algorithm in which the proposal is always accepted (Gelman et al., 2014). Combinations of the aforementioned algorithms are also widely used, e.g., MH-within-Gibbs. All MCMC algorithms (including those mentioned hereafter) require a careful examination of the convergence of the Markov chains. The MCMC diagnostics used for this purpose in **shinybrms** are outlined in section Tab “MCMC diagnostics”.

Hamiltonian Monte Carlo (HMC) (initial work and major contributions by Duane et al., 1987; Neal, 1993; MacKay, 2003; Neal, 2011) is a special MCMC algorithm which is often more efficient than other MCMC algorithms, especially in case of a high-dimensional posterior distribution and correlated parameters (Hoffman and Gelman, 2014; Betancourt, 2018). The efficiency of HMC is due to the fact that it takes advantage of the gradient of the (log) posterior density (Stan Development Team, 2022a), making it a combination of stochastic and deterministic procedures (which explains why HMC is also known as *hybrid Monte Carlo*) (Gelman et al., 2014). HMC provides helpful diagnostics, such as divergent transitions which can (but must not necessarily) indicate areas of the posterior which are hard to explore by the HMC sampler (Betancourt, 2018; Gabry et al., 2019). Compared to Gibbs sampling, HMC also has the advantage that nonconjugate priors may be used easily. For the original HMC algorithm, three tuning quantities need to be specified by hand in advance: the *mass matrix* M (which is the covariance matrix of the auxiliary momentum vector), the number L of *leapfrog steps*, and the size ϵ of the leapfrog steps (Gelman et al., 2014; Stan Development Team, 2022a).

Because of the fixed choice of L , the original HMC algorithm is a *static* HMC algorithm (Betancourt, 2018). In contrast, the *no-U-turn sampler (NUTS)* (Hoffman and Gelman, 2014) is a *dynamic* HMC algorithm since it automatically chooses a (possibly) new value of L in each iteration of each Markov chain. Hoffman and Gelman (2014) also proposed a new dual averaging technique for determining ϵ automatically, too. Apart from these automations, the NUTS has the advantage that in terms of efficiency, it was shown to perform as well as—or even better than—a well-tuned static HMC algorithm (Hoffman and Gelman, 2014). A modified (Betancourt, 2018) NUTS is implemented in Stan. Stan’s NUTS also includes an automatic adaptation of the mass matrix M during the warmup phase (Stan Development Team, 2022a). A complete presentation of Stan’s NUTS is out of the scope of this article. A good starting point for a detailed description is Stan Development Team (2022a) as well as Betancourt (2018). Note that Stan also includes other algorithms for inferring or approximating the posterior. In this paper however, we only refer to Stan’s NUTS when referring to Stan.

3 Existing GUIs

Table 1 summarizes existing GUIs for BRMs. Details are provided in Supplement section “Existing GUIs”. Table 1 makes it clear that none of the existing GUIs relies *entirely* on Stan or the NUTS. JASP does use Stan for some analyses, but JASP’s concept is quite different from **shinybrms**’s concept: While JASP offers a plenty of different statistical methods (including non-regression analyses), **shinybrms** is designed to be as concise as possible. While JASP’s approach of using Stan for only some analyses certainly has a few advantages (especially in terms of runtime), **shinybrms**’s approach of completely relying on Stan (and **brms** in particular) has the advantage of a better maintainability: **shinybrms** only

³Here, the term “proposal” refers to a proposed parameter vector.

⁴Here, the term “symmetric” refers to the preservation of the distribution when reverting the jump, not to the symmetry in the shape of a distribution.

Table 1: Existing GUIs for BRMs. Algorithmic details for the GUIs may be found in Supplement section “Existing GUIs”. “BB” stands for “Bayesian bootstrap”. Here, the term “NUTS” includes Stan’s NUTS. The column “Algorithm choice” specifies if given a model, the user may choose an algorithm (at least for some types of models). Notes: (i) The TEET package is noncommercial, but MATLAB is commercial; (ii) JASP uses (Stan’s) NUTS for Bayesian meta-analyses and mixed BRMs; (iii) For linear regression models, BEsmarter offers a choice between MCMC and the Bayesian bootstrap.

GUI name	Commercial	Algorithm (for inferring the posterior)							Algorithm choice	
		Analytic	Non-Numerical	MC	BB	Non-HMC	Static HMC	NUTS		
WinBUGS (Lunn et al., 2000)	no	no	no	no	no	yes	no	no	no	no
OpenBUGS (Spiegelhalter et al., 2014)	no	no	no	no	no	yes	yes	no	no	no
BugsXLA (Woodward, 2011)	no	no	no	no	no	yes	yes	no	no	no
IBM SPSS Amos (Arbuckle, 2020)	yes	no	no	no	no	yes	yes	no	no	yes
TEET (Qian, 2011)	no ⁽ⁱ⁾	yes	yes	yes	no	yes	no	no	no	no
JASP (JASP Team, 2022)	no	yes	yes	yes	no	yes	no	yes ⁽ⁱⁱ⁾	no	no
BRNPM (Karabatsos, 2015, 2017)	no	no	no	no	no	yes	no	no	no	no
Stata (StataCorp, 2019b)	yes	no	no	no	no	yes	no	no	no	yes
BayES (Emvalomatis, 2020)	no	no	no	no	no	yes	no	no	no	no
IBM SPSS (IBM Corp., 2020)	yes	yes	yes	yes	no	no	no	no	no	no
BEsmarter (BEsmarter Team, 2020a,b; Ramírez-Hassan and Graciano-Londoño, 2021)	no	no	no	no	yes	yes	yes	no	no	yes ⁽ⁱⁱⁱ⁾

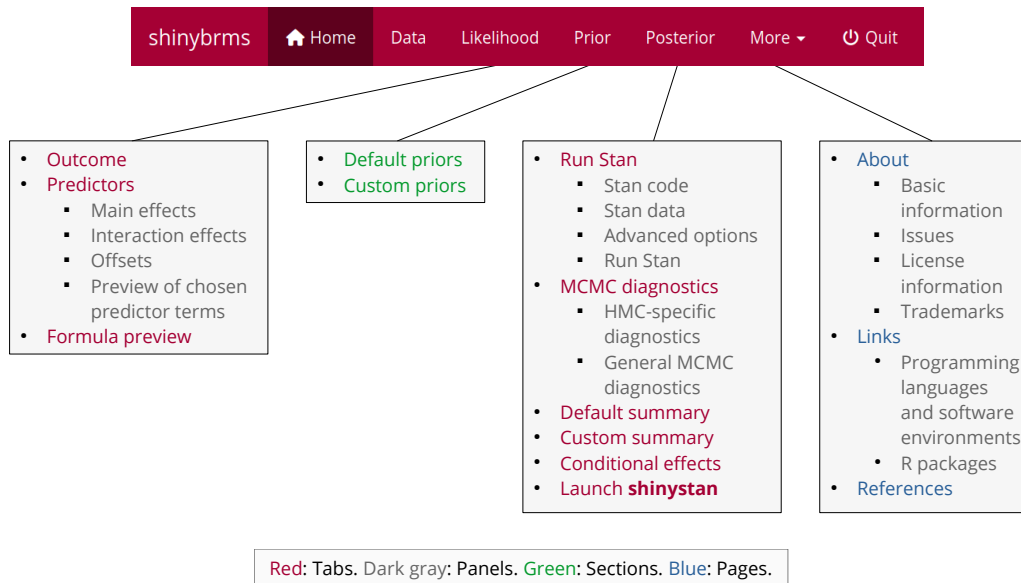


Figure 1: The navigation bar in shinybrms. In this figure, we have expanded the navigation bar itself by the structure of the three main pages (“Likelihood”, “Prior”, and “Posterior”) as well as of the drop-down menu “More”.

provides a lightweight GUI and only needs to perform few computations on its own. This is due to brms which is very flexible by allowing to fit a variety of regression models within a single R package. This division of work between shinybrms, shiny, brms, rstan, Stan, and shinystan reduces the amount of maintenance necessary for shinybrms, resulting in a faster integration of new features, a faster elimination of bugs, and a longer life cycle. Furthermore, it allows the authors of each component to focus on their strengths.

4 Features of shinybrms

The following general presentation of shinybrms’s features will be in written form, but with links to the corresponding screenshots from the example in section Example. In this article, not all aspects of the shinybrms app are shown in screenshots. For more screenshots, see the shinybrms website (Weber, 2022).

Note that the mathematical formulation of the models which may be fit with shinybrms has already been given elsewhere (Bürkner, 2017, 2018), so we will keep it short here.

Overview

The shinybrms app has three main pages which are accessible from a navigation bar at the top (Figure 1): “Likelihood”, “Prior”, and “Posterior”. This structure follows Bayes’ theorem, simplified to the proportionality of the posterior density to the product of prior density and likelihood:

$$p(\theta|\mathcal{D}) \propto p(\theta) \cdot p(\dot{\mathcal{D}}|\theta, \ddot{\mathcal{D}}) \tag{2}$$

where we have split up the data \mathcal{D} into $\mathcal{D} = [\dot{\mathcal{D}} \ \ddot{\mathcal{D}}]$ because in BRMs, the distribution in the likelihood typically conditions on the predictor part $\dot{\mathcal{D}}$ of the data (see section Tab “Predictors” below). In the following sections, these three main pages will be described in detail.

There are also some auxiliary pages, the first two having direct links in the navigation bar, the last three being accessible from the drop-down menu “More” at the end of the navigation bar:

- The starting page “Home” gives a short overview of shinybrms’s objective and structure. Thus, it only contains informational text and no interactive elements.
- On page “Data”, the user uploads his or her custom dataset which shall be used for the regression analysis. For testing purposes, page “Data” also offers example datasets. The chosen dataset (no matter if it was uploaded or chosen from the list of example datasets) is automatically shown in a preview consisting of the dataset’s first six rows (there is an option to show the full dataset,

though). It is also possible to show the output of R's `str()` function applied to the chosen dataset, which gives some basic information about the dataset and its variables for users familiar with R.

- Page “About” contains basic information about **shinybrms** (e.g., version and corresponding date) as well as some legal information.
- Page “Links” gives links to software relevant for the **shinybrms** app.
- Page “References” contains the references for literature cited throughout the app.

Page “Likelihood”

Page “Likelihood” has three tabs: “Outcome”, “Predictors”, and “Formula preview”. These tabs will now be described in turn.

Tab “Outcome”

On tab “Outcome” (Figure 2), the user specifies the outcome variable $\mathbf{y} = (y_1, \dots, y_N)^\top \in \mathbb{R}^N$ (by choosing it from a drop-down list of the variables present in the dataset) as well as its distributional family, i.e., the basic form of the likelihood $p(\mathcal{D}|\theta, \mathcal{D})$, now with $\mathcal{D} = \mathbf{y}$. For the distributional family, there is a drop-down menu and a checkbox called “Show advanced distributional families”. By default, this checkbox is unchecked which means that the drop-down menu offers three general distributional families of broad practical relevance: the Gaussian family (with the identity link function), the Bernoulli family with the logit link function, and the negative binomial family with the log link function. This is intended to be a limited selection: By reducing the choices as much as possible, we want to avoid overwhelming the user with a variety of special distributions. For example, the Poisson family is intentionally left out, in favor of the more general negative binomial distribution. However, by checking the “Show advanced distributional families” checkbox, the drop-down menu is extended so that a variety of other distributional families can be selected as well (see Supplement section “Advanced distributional families”).

Tab “Predictors”

If desired, the user may specify predictors on tab “Predictors” (Figure 3). We use the term “predictor” for a column in the model matrix. In contrast, we use the term “predictor variable” for a column in the input dataset. Thus, a predictor may also denote an interaction and a predictor variable with K categories leads to $K - 1$ predictors (due to dummy coding).

The **shinybrms** app supports population-level effects as well as group-level effects⁵. The inclusion of group-level effects yields a multilevel model (also known as hierarchical or mixed-effects model). Here, we denote the vector of population-level effects by $\boldsymbol{\beta}$ and the corresponding model matrix by \mathbf{X} . Likewise, we denote the vector of group-level effects by \mathbf{u} and the corresponding model matrix by \mathbf{Z} . The hyperparameters for the group-level effects (i.e., their standard deviations and correlations) will be collected in a vector $\boldsymbol{\tau}$. If the model does not contain group-level effects, we define here (for the mathematical description, not for the software) $\mathbf{u} = 0$, $\mathbf{Z} = (0, \dots, 0)^\top \in \mathbb{R}^N$ (for example; the exact values in \mathbf{Z} do not matter if $\mathbf{u} = 0$), and $\boldsymbol{\tau} = 0$ (for example). With \mathbf{X} and \mathbf{Z} , we now have $\tilde{\mathcal{D}} = [\mathbf{X} \ \mathbf{Z}]$. We denote the vector of linear predictors by $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} \in \mathbb{R}^N$ (written out as $\boldsymbol{\eta} = (\eta_1, \dots, \eta_N)^\top$).

Note that here as well as in **shinybrms**, the term “interaction” is also used for interactions involving predictor variables with group-level main effects (yielding group-level interaction effects). This broad definition of “interaction” simplifies the GUI and emphasizes the key concept of interactions, namely that an effect depends on another predictor (or on other predictors).

To avoid common mistakes, **shinybrms** imposes some restrictions: Firstly, an overall (population-level) intercept is always included. Secondly, including an interaction causes all corresponding lower-order interactions to be automatically included, too. The latter restriction also implies that interactions may only involve predictor variables for which main effects have already been added.

⁵Population-level effects are also known as fixed effects (Bürkner, 2017, 2018). Group-level effects are also known as random or partially pooled effects (Bürkner, 2017, 2018; Goodrich et al., 2022). The terms “fixed” and “random” effects are not really appropriate in a Bayesian context: In a Bayesian model, all parameters have a prior distribution and may therefore be considered as random (Marchenko and Balov, 2015).

Tab “Formula preview”

The tab “Formula preview” simply combines the chosen outcome and the chosen predictors into **brms**’s formula syntax. This is mainly intended for checking the correct specification of the model formula (for users familiar with the syntax). However, it also provides a concise and standardized way to communicate this central part of the model because together with the distributional family, this model formula determines the likelihood $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{D}) = p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}, \mathbf{Z})$: In case of the Gaussian family (with the identity link function), we have

$$p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}, \mathbf{Z}) = \prod_{i=1}^N \frac{1}{\sqrt{2\pi} \cdot \sigma} \exp\left(-\frac{1}{2} \cdot \left(\frac{y_i - \mu_i}{\sigma}\right)^2\right) \quad (3)$$

with $\mu_i = \eta_i$ (see $\boldsymbol{\eta}$ from section Tab “Predictors”), $\sigma \in (0, \infty)$, and $\boldsymbol{\theta} = (\boldsymbol{\beta}^\top, \mathbf{u}^\top, \boldsymbol{\tau}^\top, \sigma)^\top$. Note that the dependence on \mathbf{X} and \mathbf{Z} as well as on most elements of $\boldsymbol{\theta}$ is an indirect dependence *via* μ_1, \dots, μ_N . In case of the Bernoulli family with the logit link function, we have $\mathbf{y} \in \{0, 1\}^N$ and

$$p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}, \mathbf{Z}) = \prod_{i=1}^N \mu_i^{y_i} (1 - \mu_i)^{(1-y_i)} \quad (4)$$

with $\mu_i = \frac{1}{1 + \exp(-\eta_i)}$ and $\boldsymbol{\theta} = (\boldsymbol{\beta}^\top, \mathbf{u}^\top, \boldsymbol{\tau}^\top)^\top$. In case of the negative binomial family with the log link function, we have $\mathbf{y} \in (\{0\} \cup \mathbb{N})^N$ and

$$p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}, \mathbf{Z}) = \prod_{i=1}^N \binom{y_i + \zeta - 1}{y_i} \left(\frac{\mu_i}{\mu_i + \zeta}\right)^{y_i} \left(\frac{\zeta}{\mu_i + \zeta}\right)^\zeta \quad (5)$$

with $\mu_i = \exp(\eta_i)$, $\zeta \in (0, \infty)$, and $\boldsymbol{\theta} = (\boldsymbol{\beta}^\top, \mathbf{u}^\top, \boldsymbol{\tau}^\top, \zeta)^\top$. The mathematical details of the “advanced” distributional families (see section Tab “Outcome” above) may be found in the **brms** vignette “Parameterization of Response Distributions in **brms**”.

Page “Prior”

At the top of page “Prior”, the user obtains a preview of the default priors taken from **brms** (Figure 4). At the bottom, the user may specify custom priors (Figure 5). Both, the default and the custom priors, refer to the parameters of the currently specified likelihood. Custom priors may be specified as follows:

- *via* a Stan function,
- *via* one of the special **brms** (pseudo-)functions designed for this purpose, e.g., for the Lewandowski-Kurowicka-Joe (LKJ) prior (Lewandowski et al., 2009),
- *via* an empty input field to specify a flat prior over the whole support of the corresponding parameter(s).

The first two possibilities always lead to a proper prior. The flat prior is only proper if the support is bounded on both sides. Otherwise (which is the more common case), the flat prior is improper.

The user’s selections on page “Prior” ultimately lead to the specification of $p(\boldsymbol{\theta})$. Together with the likelihood, this completes the model specification. Note that for multilevel models, $p(\boldsymbol{\theta})$ here⁶ includes the distributions of the group-level effects \mathbf{u} as well, even though they do not need to be specified on page “Prior”.

Page “Posterior”

Page “Posterior” has six tabs: “Run Stan”, “MCMC diagnostics”, “Default summary”, “Custom summary”, “Conditional effects”, and “Launch **shinystan**”. These will now be described in turn. The output shown on these tabs may also be downloaded (with the file format depending on the specific type of output).

Tab “Run Stan”

At the top of tab “Run Stan”, the user may inspect and download the Stan code and the so-called *Stan data*. The Stan data basically consists of the pre-processed part of the chosen dataset which is

⁶In multilevel models, drawing the line between prior and likelihood can be done in multiple ways, so our mathematical formulation is just one of several possibilities.

needed for the Stan model, extended by some internal objects. Apart from checking or documentation purposes, the Stan code and the Stan data are needed if the user wants to customize the Stan code and then run Stan outside of **shinybrms**.

Further down on tab “Run Stan”, the user may set advanced options for the Stan run (Figure 6). These options have sensible defaults, but sometimes they need to be changed. Probably the most important option is the seed for the pseudorandom number generator.

The final panel on tab “Run Stan” is the central one (Figure 7): By a click on the “Run Stan” button, Stan translates the Stan code written by **brms** to C++ code, compiles this C++ code, and then starts sampling. By default, Stan writes its sampling progress to an HTML file which is automatically opened up by **shinybrms**. The user then only needs to refresh this HTML file to see the current sampling progress. An example for Stan’s runtime will be given in section [Example](#).

When Stan has finished sampling, the panel “Run Stan” automatically refreshes, in particular to show the result from an overall check of the MCMC diagnostics (see section [Tab "MCMC diagnostics"](#) for details). The user also has the possibility to download different output objects which can be analyzed outside of **shinybrms** or—in case of the fitted model object of class “`brmsfit`”—uploaded in a new **shinybrms** session (to avoid re-running Stan).

Tab “MCMC diagnostics”

On tab “MCMC diagnostics” (Figure 8), the user obtains detailed information concerning the following MCMC diagnostics:

- HMC-specific diagnostics:
 - the number of iterations ending with a divergence,
 - the number of iterations hitting the maximum tree depth,
 - the Bayesian fraction of missing information for the energy transitions (E-BFMI);
- some general MCMC diagnostics (which are computed for each parameter as well as for the accumulated log posterior density):
 - the modified potential scale reduction factor \hat{R} proposed by [Vehtari et al. \(2021\)](#) (here simply called *the \hat{R}* instead of *the modified \hat{R}*)⁷,
 - the effective sample size (ESS) in the bulk of the corresponding marginal posterior ([Vehtari et al., 2021](#)),
 - the ESS in the tails of the corresponding marginal posterior ([Vehtari et al., 2021](#)).

As a full description of these MCMC diagnostics is out of the scope of this article, we refer the interested reader to [Stan Development Team \(2022c\)](#), [Betancourt \(2018\)](#), and [Vehtari et al. \(2021\)](#). The most important basic guidelines for deciding whether these MCMC diagnostics are worrying are explained in the **shinybrms** GUI and also checked automatically by **shinybrms**. For the general MCMC diagnostics, it is also possible to show a detailed table with the diagnostics for each parameter (as well as for the accumulated log posterior density).

Tab “Default summary”

Tab “Default summary” (Figure 9) shows **brms**’s standard *robust* summary of the posterior inference, e.g., the medians and the central 95 % intervals of the marginal posteriors (the 95 % CrIs) of the most important parameters. This tab is only intended for a quick inspection. A much more comprehensive analysis of the Stan output is offered by the **shinystan** app (see section [Tab "Launch shinystan"](#)).

Tab “Custom summary”

On tab “Custom summary” (Figure 10), the user may calculate posterior summary quantities for a custom mathematical (or logical) expression involving at least one parameter. Such an expression may be, e.g., a sum of two parameters (as shown in Figure 10) or the event that a parameter exceeds a certain threshold.

⁷The term “potential scale reduction factor” is not always appropriate ([Vehtari et al., 2021](#), section 2), but because of its widespread use, we employ it here nonetheless.

Tab “Conditional effects”

On tab “Conditional effects” (Figure 11), **shinybrms** offers *conditional-effects plots* (created by `brms::conditional_effects()`). A conditional-effects plot shows the estimated effect of a predictor variable on the outcome. An interaction effect involving at most two predictor variables may also be visualized by showing the estimated effect of the first predictor variable separately for appropriate values of the second predictor variable.

As described in more detail in the **shinybrms** GUI, a conditional-effects plot *conditions* on specific values of those predictor variables which are not involved in the plot. Likewise, group-level effects which are not involved in the plot are (usually) set to zero.

Tab “Launch shinystan”

The **shinystan** app (Gabry, 2022) offers an interactive inspection of Stan (and other MCMC-generated) results, in particular with respect to:

- MCMC diagnostics (including several additional diagnostics not covered by **shinybrms**’s tab “MCMC diagnostics”),
- PPCs,
- summary quantities of univariate marginal posteriors,
- plots of univariate, bivariate, and trivariate marginal posteriors.

Before launching the **shinystan** app from within the **shinybrms** app by clicking the corresponding button on tab “Launch **shinystan**”, the user may set a seed to ensure the reproducibility of the PPCs.

At this point, the **shinybrms** workflow ends and passes over to the **shinystan** workflow. We will illustrate the **shinystan** workflow in section [Example](#).

5 Example

We illustrate **shinybrms**’s features following the workflow implied by Figure 1 and using a real-world dermatological dataset from Van Welzen et al. (2021). This dataset is available in the Supplement (file ‘CAP.csv’). In Supplement section “Frequentist analysis of the example”, we compare the Bayesian analysis presented here with a frequentist one, referring to the list of advantages of Bayesian statistics from section [Introduction](#).

Van Welzen et al. (2021) conducted a prospective pilot study investigating the efficacy and safety of a novel cold atmospheric plasma (CAP) wound dressing for the healing of split-skin graft donor sites. The only outcome we focus on here is the tissue water index (TWI), measured by a hyperspectral imaging camera and having values between 0 and 100, with lower TWI values being associated with an improved wound healing. Briefly, the study design was as follows: For each of $P = 10$ patients, the TWI was measured under $T = 3$ different treatment conditions (standard-treated wound, CAP-treated wound, and healthy skin). Each treatment condition was investigated in its own skin area with $R = 3$ measurements across that area. This procedure of measuring was repeated on each of $D = 4$ days (day 1, 3, 5, and 7, with day 1 being the day of the split-skin graft donation where the TWI was measured *after* the split-skin graft donation but *before* the first wound dressing). Thus, the dataset consists of $N = P \cdot T \cdot R \cdot D = 360$ observations (rows). The dataset’s columns are:

- patID (for “patient ID”; coded as “pat1”, ..., “pat10”),
- age (in years),
- anticoagulation (indicating whether the patient received an anticoagulation therapy before and during the study; coded as “no” and “yes”),
- diabetes (indicating whether the patient is diabetic; coded as “no” and “yes”),
- day (coded as “d1”, “d3”, “d5”, and “d7”),
- trt (coded as “0_standard”, “CAP”, and “healthy” to make “0_standard” the reference level),
- TWI (integers in the interval $[0, 100]$).

The primary research question is whether the CAP treatment leads to a decreased⁸ TWI compared to the standard treatment (polyhexanide wound gel with fatty gauze). The healthy skin area serves as an *experimental* control (albeit not as a control *treatment* since this is the role of the standard-treated wound area) and is not part of the primary research question.

⁸For this demonstration here, we won’t discuss whether this decrease is clinically relevant.

The screenshot shows the 'Likelihood' tab in the 'shinybrms' application. On the left, a sidebar contains three tabs: 'Outcome' (selected), 'Predictors', and 'Formula preview'. The main content area is titled 'Outcome' and contains the following elements:

- A heading: 'Outcome'
- Text: 'Choose the outcome (the dependent variable) and the distributional family for this outcome.'
- 'Outcome:' dropdown menu: 'TWI'
- 'Distributional family for the outcome:' dropdown menu: 'Gaussian (normal)'
- Checkbox: 'Show advanced distributional families' (unchecked)
- Text: 'Parameters specific to this distributional family, with their link function as used in shinybrms:'
- Table:

Parameter	Link function
mu	identity
sigma	log
- Text: 'For details concerning the link functions, see the help for the R function `brms::brmsfamily()` and the `brms` vignette "Parameterization of Response Distributions in `brms`". Note that for each parameter, the link function only applies if this parameter is actually modeled by (nonconstant) predictors. In `shinybrms`, this is currently only supported for the location parameter `mu`.'
- Text: 'For details concerning the remaining (family-specific) parameters, see the help for the R function `brms::set_prior()`.'

Figure 2: Tab “Outcome” on page “Likelihood”. In the example presented here, we select TWI as the outcome variable and the Gaussian family as the distributional family for this outcome. Tab “Outcome” and tab “Predictors” (Figure 3) are the two main components of page “Likelihood”.

shinybrms

After launching the `shinybrms` app in R *via*

```
> library("shinybrms")
> launch_shinybrms(launch.browser = TRUE)
```

(with `launch.browser` set to `TRUE` to ensure that the app is opened up in the default web browser), we switch to page “Data” where we upload the dataset (not shown here).

Next, we head over to page “Likelihood”. On tab “Outcome” (Figure 2), we choose the outcome variable TWI and the Gaussian family as the distributional family for this outcome. Clearly, the TWI values cannot follow an unmodified Gaussian distribution since they are bounded by 0 and 100, with the minimum of the observed TWI values being indeed as low as 9 (the maximum being 67). Thus, a truncated Gaussian distribution might be more appropriate here. We will come back to this later in section [Discussion](#).

On tab “Predictors” (Figure 3), we choose age, anticoagulation, diabetes, day, and `trt` to have population-level main effects and `patID` to have group-level main effects (“random intercepts”). Further down on tab “Predictors”, we add an interaction between day and `trt` (not shown here). This interaction is included because the TWI is supposed to show a stronger time-dependence in the two wound areas than in the healthy skin area. Additionally, the TWI difference (in means) between the standard and the CAP treatment might change over time.

Now the likelihood is set up, so we can proceed with the prior. The default priors (Figure 4) are reasonable, but suppose we wanted a weakly informative Student-*t* prior with 3 degrees of freedom, a location parameter of 0, and a scale parameter of 30 for all regression coefficients. To add this custom prior, we choose parameter class `b` from the corresponding drop-down list shown in Figure 5, enter `student_t(3, 0, 30)` into the input field entitled “Prior distribution”, and click the “Add prior” button. After doing so, our Student-*t* prior is added to the preview table (Figure 5, right-hand side). For all remaining parameters for which we do not specify a custom prior, the corresponding default prior will be used.

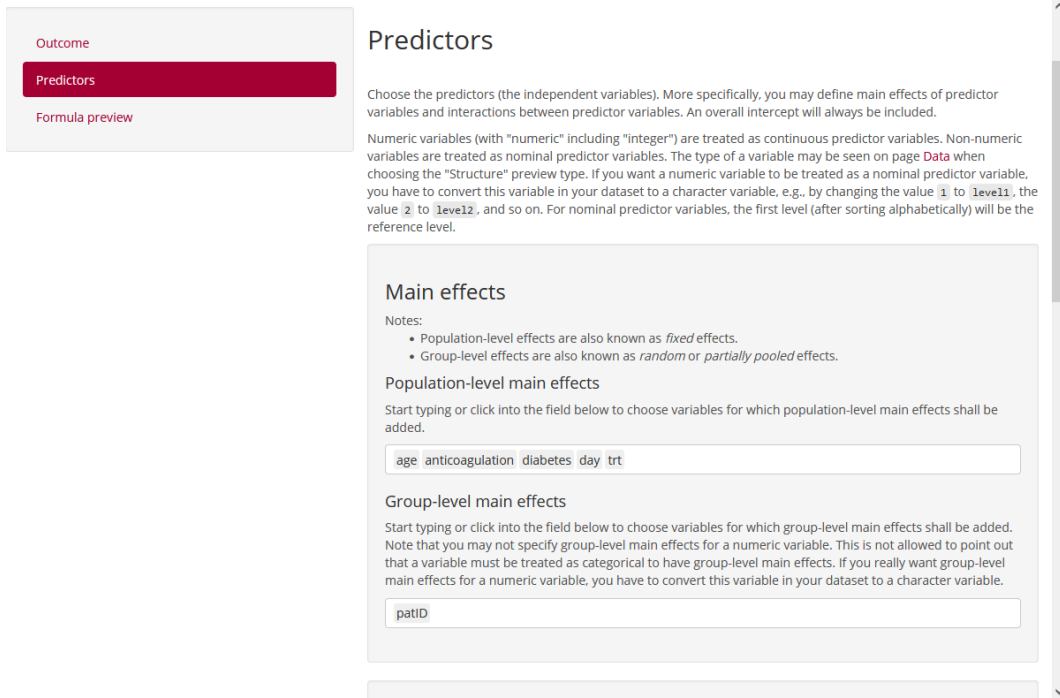


Figure 3: Tab “Predictors” on page “Likelihood”. Here, the main effects of the predictors need to be defined first (in the example presented here: variables age, anticoagulation, diabetes, day, trt, and patID). Then, further down on this tab (not visible here), interactions can be specified (in the example presented here: an interaction between variables day and trt). In principle, offsets may also be specified further down on this tab (not visible here), but our example does not feature offsets. For the main effects, the user may choose between population-level and group-level effects. For interaction effects, this choice will be performed automatically based on the involved main effects.

most recent version of **brms**, the appropriate version depends on the installed **brms** version.

Default priors

Default priors for the parameters belonging to the current likelihood:

Prior	Class	Coefficient	Group	Lower bound	Upper bound
(flat)	b				
(flat)	b	age			
(flat)	b	anticoagulationyes			
(flat)	b	dayd3			
(flat)	b	dayd3:trtCAP			
(flat)	b	dayd3:trthealthy			
(flat)	b	dayd5			
(flat)	b	dayd5:trtCAP			
(flat)	b	dayd5:trthealthy			
(flat)	b	dayd7			
(flat)	b	dayd7:trtCAP			
(flat)	b	dayd7:trthealthy			
(flat)	b	diabetesyes			
(flat)	b	trtCAP			
(flat)	b	trthealthy			
student_t(3, 42, 7.4)	Intercept				
student_t(3, 0, 7.4)	sd			0	
student_t(3, 0, 7.4)	sd		patID	0	
student_t(3, 0, 7.4)	sd	Intercept	patID	0	
student_t(3, 0, 7.4)	sigma			0	

Figure 4: Section “Default priors” on page “Prior”. The default priors are taken from **brms** and depend on the currently specified likelihood. They can be overridden by custom priors (Figure 5).

Custom priors

Specification of a custom prior

Class:
Note: This is the parameter class. It may consist of a single parameter.

b

Coefficient:
Note: Leave empty to use all coefficients belonging to the selected class.

Choose coefficient or leave empty

Group (for group-level effects):
Note: Leave empty while having an empty "Coefficient" field to use all groups belonging to the selected class.

Choose group or leave empty

Prior distribution:
Note: You may either

- specify a prior distribution using a Stan function (see the "Stan Functions Reference" for details), or
- specify a prior distribution using one of the special (pseudo-)functions defined by `brms` for this purpose (see `brms::set_prior()` for details; an important example is `lkj` for parameters of class `cor`), or
- leave this field empty to use a flat prior.

If you specify a prior distribution using a Stan function, you have to use the Stan function which would be used in a Stan sampling statement and specify values for all arguments of this Stan function (e.g., `normal(0, 2.5)`).

student_t(3, 0, 30)

Add prior

Reset all custom priors

Preview of custom priors

Custom priors currently set:

Prior	Class	Coefficient	Group
student_t(3, 0, 30)	b		

An empty field in column "Prior" denotes a flat prior over the support of the corresponding parameter.

Figure 5: Section "Custom priors" on page "Prior". Here, we specify a Student- t prior with 3 degrees of freedom, a location parameter of 0, and a scale parameter of 30 for all regression coefficients (parameter class `b`). This overrides the default flat prior for these parameters (Figure 4).

Now the model is fully set up, so we can start inferring the posterior. To do this, we switch to page "Posterior" where we scroll down to the advanced options on tab "Run Stan" (Figure 6). There, we set a seed for reproducibility. Afterwards, we scroll further down to panel "Run Stan" where we click the button for starting the Stan run (Figure 7). For this example, the Stan run as a whole (including the compilation of the C++ code) takes about 50 seconds on a standard desktop machine.

After Stan has finished sampling, we receive a pop-up notification (not shown here) whether all MCMC diagnostics have passed their checks. Here, this is the case as we may also see on tab "MCMC diagnostics" (Figure 8).

Since `shinybrms` reports all MCMC diagnostics as being OK, we may start interpreting the posterior. On tab "Default summary" (Figure 9), it is mainly the summary of the population-level effects which is of interest here: With each additional year of age, the TWI is estimated to increase by ca. 0.10 with a 95 % CrI of ca. (-0.28, 0.48). An anticoagulation therapy is estimated to increase the TWI by ca. -1.12 with a 95 % CrI of ca. (-7.53, 5.69). A diabetes disease is estimated to increase the TWI by ca. -0.56 with a 95 % CrI of ca. (-6.51, 5.66). As may be seen from these three CrIs, the statistical uncertainty is quite big which is probably due to the small $P = 10$.

Since we included an interaction between `day` and `trt`, the coefficients for these two variables are most conveniently interpreted by the help of a custom summary (Figure 10) and a conditional-effects plot (Figure 11). On tab "Custom summary" (Figure 10), we may calculate the estimated TWI difference (in means) between the CAP and the standard treatment separately for each day by entering the corresponding sum expressions (and the expression ``b_trtCAP`` for day 1) in turn. The resulting table is included in Figure 10: On day 1 (where the two wound areas had not been treated yet), the standard treatment and the CAP treatment lead to a quite similar TWI (the posterior median of their TWI difference being ca. -0.74 with a 95 % CrI of ca. (-3.69, 2.13)). In contrast, on days 3, 5, and 7, the CAP treatment clearly leads to a *lower* TWI than the standard treatment (posterior medians of ca. -10.45, -7.66, and -7.10, respectively, and 95 % CrIs of ca. (-13.43, -7.36), (-10.63, -4.84), and (-10.11, -4.03), respectively). This answers the primary research question: The CAP treatment indeed leads to a decreased TWI and therefore an improved wound healing compared to the standard treatment. This is also well illustrated by the conditional-effects plot (Figure 11). The conditional-effects plot also confirms that the TWI in the healthy skin area does not change as heavily over time as in the two wound areas.

Advanced options

Here, you can set advanced options for the R function `brms::brm()` which is the central function for inferring the posterior. These advanced options have sensible defaults, but sometimes, they need to be changed.

Show advanced options

For most of the following advanced options, details may be found on the `brms::brm()` help page. However, there are also some backend-specific advanced options for which the following help pages need to be consulted:

- For the **rstan** backend: `rstan::sampling()`, together with `rstan::stan()`.
- For the **cmdstanr** backend: `cmdstanr::sample()`, together with the "[CmdStan User's Guide](#)".

Notes:

- Numeric options with a preset value may not be left empty.
- If unset, option "Seed" internally defaults to a random seed, giving nonreproducible results. To obtain reproducible results, you need to specify a value for option "Seed" and enter this value each time you want to obtain the same results again.
- Internally, the value supplied to option "Cores" is cut off at the value supplied to option "Chains".
- If unset, option "Warmup iterations per chain" internally defaults to half of option "Total iterations per chain" (rounded down if this fraction is not an integer).
- If unset, option "Progress-refreshing step size" internally defaults to a tenth of option "Total iterations per chain", but at least 1.
- If unset, option "Range of random initial values in the unconstrained parameter space" internally defaults to 2.

Backend:
 rstan cmdstanr

Seed:

Cores:

Chains (MCMC chains):

Total iterations per chain:

Warmup iterations per chain:

Thinning rate:

Initial values:
 Random Zero

Range of random initial values in the unconstrained parameter space (`init_r` in rstan, `init` in cmdstanr; only relevant if random initial values are chosen):

Target Metropolis acceptance rate (`adapt_delta`):

Maximum tree depth (`max_treedepth`):

Open progress

Progress-refreshing step size (`refresh`):

Save draws for *all* parameters, including internal ones

Save warmup

Figure 6: Panel “Advanced options” on tab “Run Stan” of page “Posterior”. The defaults for these advanced options should be fine for most practical situations. In the example presented here, we only set a specific seed so that results are reproducible.

Finally, we switch to tab “Launch **shinystan**”, enter a seed for the reproducibility of the PPCs (here, 63438), and click on the button for launching **shinystan**.

shinystan

Within **shinystan**, we may inspect some PPC plots, e.g., a kernel density estimate for the observed TWI values, overlaid by kernel density estimates for replicated TWI values (Figure 12). This overlaid density plot suggests that the model is appropriate, being able to generate outcome values similar to the observed ones after having estimated the unknown parameters by the help of the observed dataset (as well as the prior). Nevertheless, the model may still be improved, as illustrated by the PPC plots shown in Figure 13 (lower two histograms): The minimum of the observed TWI values is systematically smaller than the replicated minimums, the opposite holding—even if not that extremely—for the maximum. However, we consider the current model to be appropriate for the primary research question.

With respect to the parameter estimates, **shinystan** offers, e.g., a visualization of the posterior medians, together with 50 % and 95 % CrIs (Figure 14). The **shinystan** app also offers kernel density estimates for the univariate marginal posteriors (not shown).

6 Discussion

We have presented our **shiny** app called **shinybrms**, distributed as an R package. With the **shinybrms** GUI, we hope to make Bayesian regression modeling more accessible for people without any knowledge of R’s syntax. Currently, the user still needs to execute some R code for setting up **shinybrms**’s backend and for launching the **shinybrms** app, even if he or she is using a GUI for installing R packages. We tried to make this as easy as possible by providing step-by-step instructions in the ‘README’ file of the **shinybrms** package. More importantly however, the **shinybrms** app may be hosted on a

Run Stan

Start the Stan run for inferring the posterior here (or upload the results from a previous Stan run instead).

Notes:

- If the advanced option "Open progress" is selected (as per default), Windows users having Firefox set as their default web browser may need to manually copy the link to the Stan HTML progress file which is automatically opening up and paste this link into a different web browser for viewing the progress file there.
- In general, uploading the results from a previous Stan run will cause a mismatch between the content shown on page **Posterior** versus the content shown on pages **Likelihood** and **Prior**.
- If uploaded Stan results are used, then **shinybrms** currently cannot check whether the number of chains in the Stan results differs from the desired number of chains (i.e., from the number of chains specified originally).

Run Stan (may take a while)

Upload "brmsfit" object (RDS file):

Browse ... No file selected

Date and time when the Stan run was finished:

Important software versions used for this Stan run:

Check if all MCMC diagnostics are OK (see the tab **MCMC diagnostics for details):**

Choose output file to download:

"brmsfit" object (RDS file)

The most comprehensive output object is the `brmsfit` object which is the output from the R function `brms::brm()`, the central function for inferring the posterior. Such a `brmsfit` object may be uploaded later above to avoid running Stan (for *that* model and *that* data) again.

Download output file

Figure 7: Panel "Run Stan" on tab "Run Stan" of page "Posterior". This is the central UI element: By clicking the red button, the Stan run is started, which first involves several preparation steps (including the compilation of the C++ code) and then the MCMC sampling itself.

server and accessed through a web browser, just like any other **shiny** app. In that case, the user does not need to install the **shinybrms** package or any other additional software. With the server-sided hosting, the **shinybrms** app may even be accessed from a mobile device where it is usually impossible to install any software designed for personal computers. Of course, setting up the server-sided hosting is a lot more complex than following the instructions from our 'README' file for running **shinybrms** on a local computer, but the idea is that IT departments of bigger institutions could establish the server-sided hosting (potentially adding an access control on top) and then members of that institution could access the **shinybrms** app through their web browsers.

Note that JASP offers an alternative host-client service by relying on rollApp (rollApp, Inc., 2020; rollApp, Inc. and JASP Team, 2020).

Apart from application in practice, **shinybrms** may also be valuable for teaching Bayesian regression models, e.g., to undergraduate students.

Of course, **shinybrms** may still be extended. As may be seen from our real-world example in section **Example**, truncated outcome families would be a useful feature. Apart from this, our future plans also include further outcome families supported by **brms** (e.g., ordinal and time-to-event regression), model selection features (e.g., using the package **projpred** by Piironen et al., 2022), and support for special **brms** features such as smoothed effects and known measurement error in the outcome variable (needed for meta-analyses). When implementing new features, the challenge will be to keep the GUI as simple as possible: In our opinion, a GUI such as **shinybrms** should support the user by automizing steps wherever this is appropriate and thus focus the attention to steps which may not be automized (in particular those related to the original research question).

diagnostics very carefully. In particular, you need to check the HMC-specific diagnostics as well as the detailed table of the general MCMC diagnostics.

HMC-specific diagnostics

Divergences:
The number of iterations ending with a divergence (0) is OK.

Hits of maximum tree depth:
The number of iterations hitting the maximum tree depth (0) is OK.

E-BFMI:
The E-BFMI (chain_1: 0.8652, chain_2: 0.9719, chain_3: 0.9089, chain_4: 0.9275) is OK.

General MCMC diagnostics

\hat{R} :
All R-hat values are OK.

Bulk-ESS:
All bulk-ESS values are OK.

Tail-ESS:
All tail-ESS values are OK.

Show detailed table of the general MCMC diagnostics

Download list of MCMC diagnostics (RDS file)

Figure 8: Tab “MCMC diagnostics” on page “Posterior”. This tab presents the diagnostics from section [Tab "MCMC diagnostics"](#), applied to the user’s Stan run (for the exact values of the general MCMC diagnostics, the checkbox “Show detailed table of the general MCMC diagnostics” needs to be checked). The purpose of this tab is to obtain details about problematic MCMC diagnostics in case there are such (after the Stan run, the user always receives a notification stating if there are problematic MCMC diagnostics or not).

7 Supplementary Material

This article comes with an online Supplement which consists of the following files:

- file ‘Supplement_sections.pdf’ which is a document with the following sections:
 - “Existing GUIs”,
 - “Advanced distributional families”,
 - “Frequentist analysis of the example”;
- file ‘CAP.csv’ which contains the dataset for section [Example](#);
- file ‘weber_shinybrms.R’ which contains the R code for section [Example](#);
- file ‘weber_shinybrms_sessionInfo.txt’ which contains the original computing environment information for section [Example](#). Note that the reproducibility of Stan results depends on the machine’s hardware, so in general, our results from section [Example](#) will not be perfectly reproducible on other machines.

Notes:

- Column **Estimate** contains the posterior median.
- Column **Est.Error** contains the posterior median absolute deviation.
- Column **l-95% CI** contains the lower boundary of the 95% central posterior interval.
- Column **u-95% CI** contains the upper boundary of the 95% central posterior interval.

```

Family: gaussian
Links: mu = identity; sigma = identity
Formula: TWI ~ 1 + age + anticoagulation + diabetes + day * trt + (1 | patID)
Data: structure(list(patID = c("pat1", "pat1", "pat1", " (Number of observations: 360)
Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
      total post-warmup draws = 4000

Priors:
b ~ student_t(3, 0, 30)
Intercept ~ student_t(3, 42, 7.4)
<lower=0> sd ~ student_t(3, 0, 7.4)
<lower=0> sigma ~ student_t(3, 0, 7.4)

Group-Level Effects:
~patID (Number of levels: 10)
      Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS Tail_ESS
sd(Intercept)  3.5251   1.1133   2.0040   7.3941 1.0011    1633    2322

Population-Level Effects:
      Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS Tail_ESS
Intercept      39.8379   11.8409  12.8092  66.0254 0.9999    2403    2326
age              0.0973    0.1682  -0.2802   0.4789 1.0004    2293    2349
anticoagulationyes -1.1233   2.8565  -7.5300   5.6943 1.0020    2490    2137
diabetesyes      -0.5574   2.7734  -6.5113   5.6622 1.0021    2925    2313
day3            -8.8831   1.5232 -11.8097  -5.8128 1.0023    2454    2777
day5            -6.6897   1.5189  -9.6449  -3.7360 1.0014    2195    2650
day7            -3.3499   1.5404  -6.5380  -0.2600 1.0020    2223    2637
trtCAP          -0.7363   1.4965  -3.6947   2.1297 1.0011    2091    2731
trthealthy      -0.5151   1.5094  -3.5835   2.4313 1.0020    2045    2433
day3:trtCAP    -9.6957   2.1596 -13.9173  -5.6282 1.0007    2571    2953
day5:trtCAP    -6.9618   2.1127 -11.1129  -2.9358 1.0020    2452    2775
day7:trtCAP    -6.3835   2.2076 -10.5728  -2.0945 1.0009    2351    2789
day3:trthealthy  9.0848   2.2005   4.8822  13.3369 1.0012    2601    2694
day5:trthealthy  7.9364   2.1746   3.7318  12.1437 1.0015    2358    2683
day7:trthealthy  8.7364   2.1793   4.5422  12.9790 1.0011    2407    2781

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS Tail_ESS
sigma  5.8362   0.2239   5.4104   6.3176 1.0009    5234    2849

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
    
```

[Download default summary \(text file\)](#)

Figure 9: Tab “Default summary” on page “Posterior”. Presented is the output of method `brms::summary.brmsfit()` with arguments `priors` and `robust` set to `TRUE` (causing the priors to be shown, too, and the more robust summary quantities median and median absolute deviation to be used instead of the less robust quantities mean and standard deviation). This tab is only intended for a quick inspection (the `shinystan` app offers a more comprehensive output).

Run Stan

MCMC diagnostics

Default summary

Custom summary

Conditional effects

Launch *shinystan*

Custom summary

Here, you may calculate posterior summary quantities for a custom mathematical (or logical) expression involving at least one parameter. Click [here](#) for a list of characters and character groups which are allowed in the custom expression. For details on how to use these characters or character groups, see the examples below or [these links](#). Parameter names need to be enclosed in backticks (` `). The drop-down list below may be used for inserting parameter names (directly with enclosing backticks) into the custom expression.

Fictitious examples for a custom expression would be:

- ``b_age` + `b_age:genderM``
- `log(`sigma`)`
- ``b_treatment` > 0.2`

For the latter example, the posterior mean gives the posterior probability that ``b_treatment` > 0.2`.

Custom expression involving at least one parameter:

``b_trtCAP` + `b_day7:trtCAP``

Parameter name to insert:

b_day7:trtCAP

Insert parameter name

Name for the custom expression (optional):

CAP_d7

Calculate posterior summary quantities

Posterior summary quantities:

Name	Q2.5	Q25	median	Q75	Q97.5	MAD	mean	SD
CAP_d1	-3.69	-1.69	-0.74	0.32	2.13	1.50	-0.71	1.52
CAP_d3	-13.43	-11.48	-10.45	-9.37	-7.36	1.55	-10.44	1.53
CAP_d5	-10.63	-8.65	-7.66	-6.66	-4.84	1.48	-7.68	1.48
CAP_d7	-10.11	-8.10	-7.10	-6.09	-4.03	1.49	-7.10	1.54

Note: All columns contain *posterior* summary quantities. In particular, the columns starting with "Q" contain the corresponding posterior percentiles and column "MAD" contains the posterior median absolute deviation.

Download custom summary (CSV file)

Figure 10: Tab “Custom summary” on page “Posterior”. In contrast to tab “Default summary” (Figure 9), users can request their own summary quantities here. In the example presented here, we calculate the day-specific CAP effects. These show that apart from day 1 (where the wound areas had not been treated yet), the CAP treatment leads to a lower (i.e., better) TWI compared to the standard treatment (which is the reference category), with the posterior median ranging from ca. -10.45 to ca. -7.10 on days 3, 5, and 7.

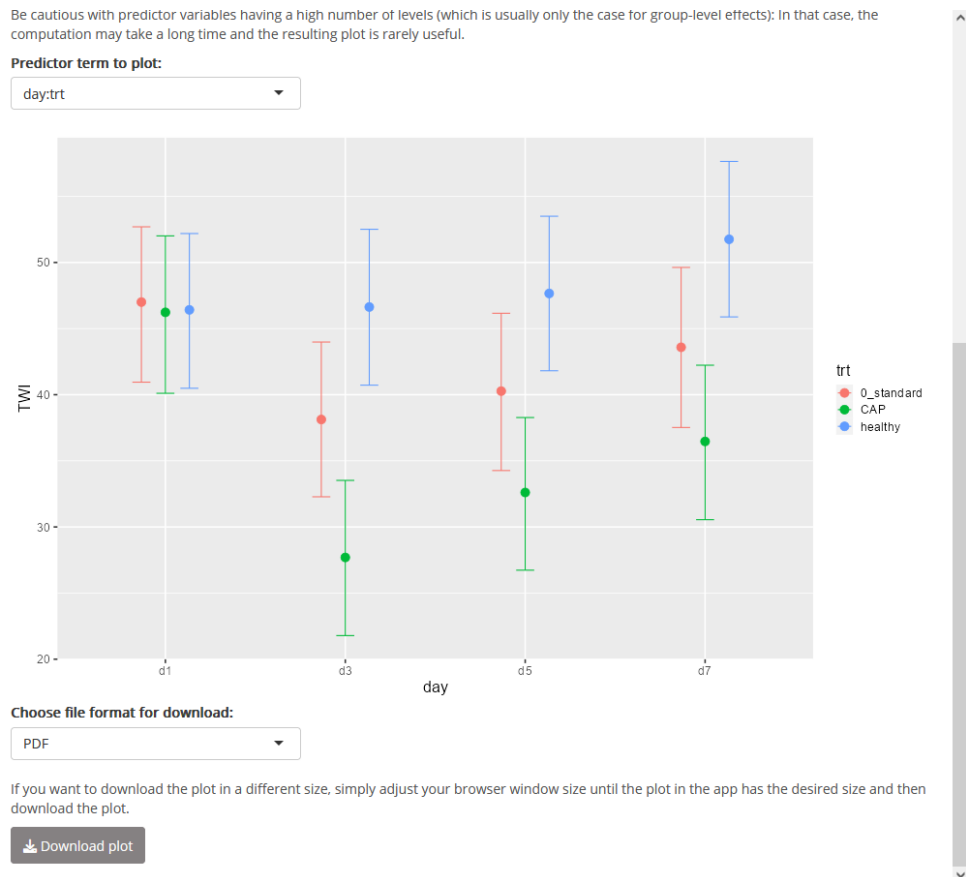


Figure 11: Tab “Conditional effects” on page “Posterior”. This tab shows the conditional-effects plots produced by `brms::conditional_effects()`. In the example presented here, we select the conditional-effects plot for the `day:trt` interaction. Similarly to Figure 10, this demonstrates that apart from day 1, the CAP treatment leads to a lower TWI compared to the standard treatment. This plot also illustrates that in the healthy skin area, the TWI is roughly constant over time, with a slight increase on day 7.

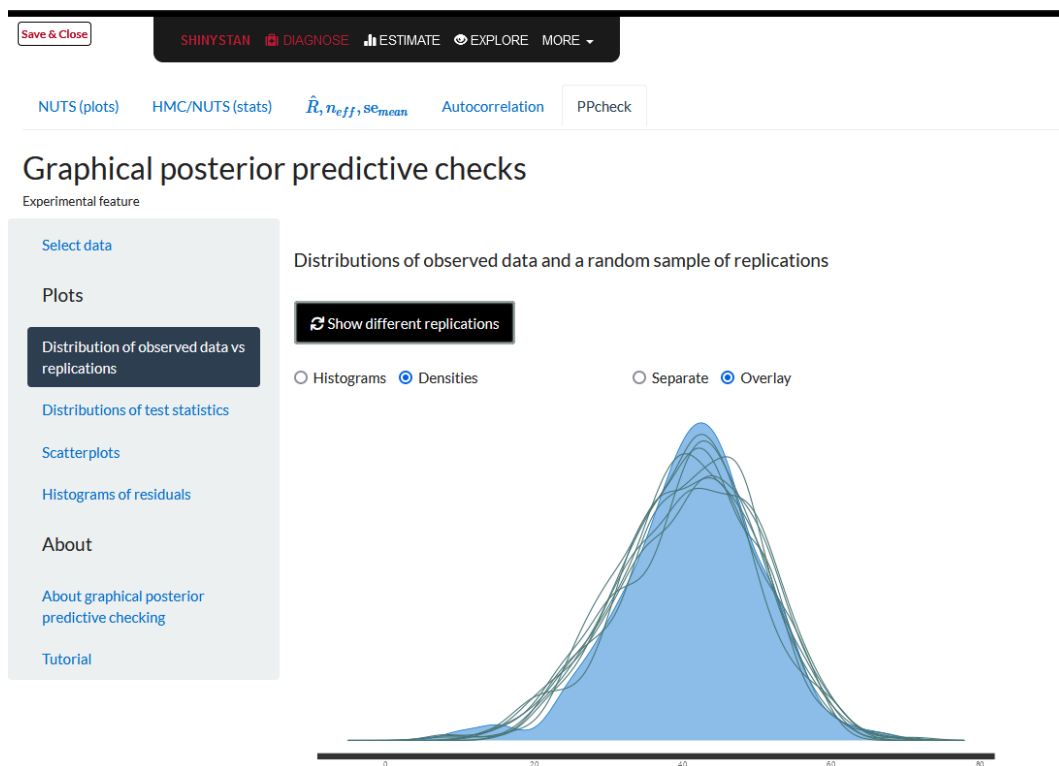


Figure 12: shinystan: PPC *via* overlaid kernel density estimates. The shaded blue density corresponds to the observed outcome values whereas each of the 8 overlaid green density lines corresponds to one randomly chosen post-warmup MCMC iteration. Here, the distributions of the model's predictions (which are based on the posterior, i.e., on the joint parameter distribution inferred from the data and the prior) are similar to the distribution of the observed outcome values, showing that at least in this regard, the model is a reasonable one.

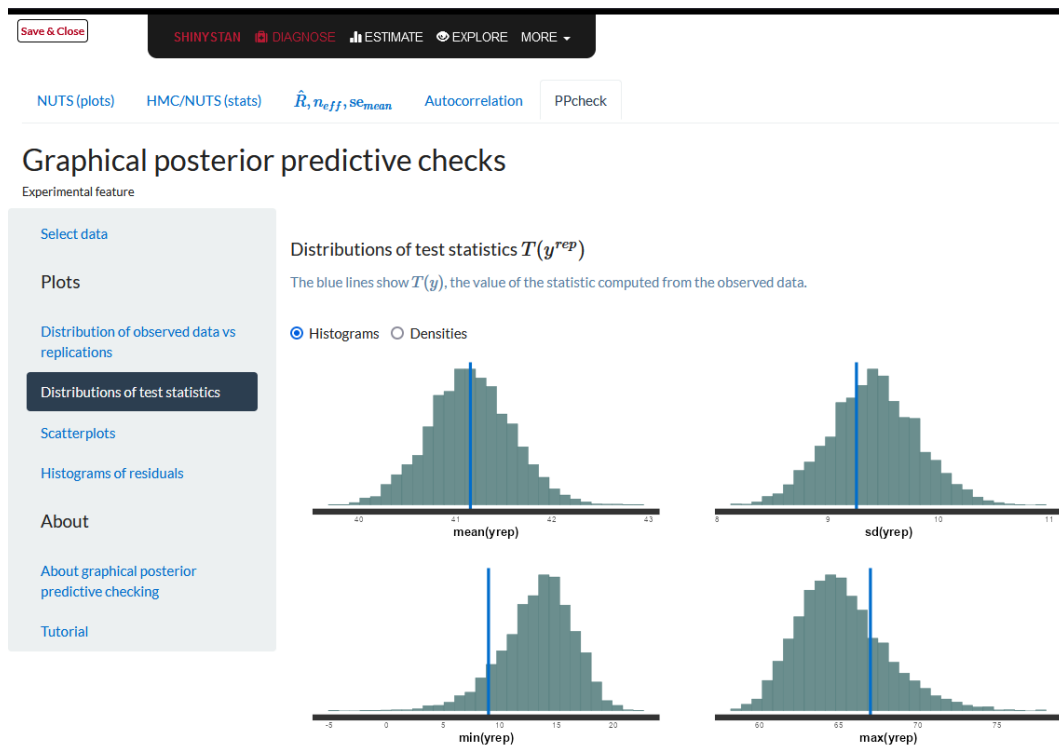


Figure 13: shinystan: PPCs *via* summary statistics. In contrast to the PPC from Figure 12, these PPCs here are based on *all* posterior draws which is possible by aggregating across the observations. The aggregation statistics are the mean (top left), the standard deviation (top right), the minimum (bottom left), and the maximum (bottom right). Here, these aggregated predictions show some room for model improvement: The minimum is overestimated—or rather “overpredicted”—by the model, the maximum is underestimated. Thus, the range of the replicated outcome values is narrower than the observed one. In contrast, the mean TWI is replicated reliably. The standard deviation shows a slight overestimation by the model. In summary, the Gaussian family seems to be a suboptimal outcome family, but we consider it to be sufficient for answering the primary research question (the comparison of CAP and standard treatment in terms of the central tendency of TWI values).

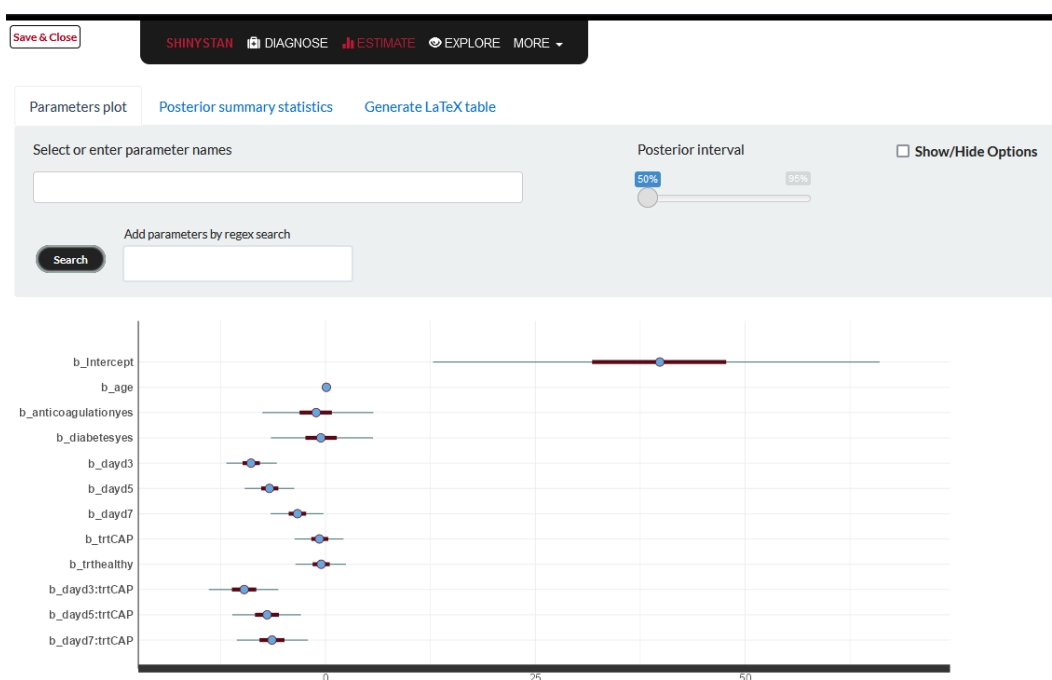


Figure 14: shinystan: Posterior intervals (credible intervals, CrIs). The default plot shown here is restricted to the first 12 parameters. More parameters may be selected in the two input fields above the plot. The different scales of the parameters (in particular, the intercept and the regression coefficient for age are on strikingly different scales) illustrate that in interactive use, it often makes sense to customize the selection of parameters.

Bibliography

- J. Albert and J. Hu. *Probability and Bayesian Modeling*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, Boca Raton, FL, USA, 2019. ISBN 978-1-351-03014-4. URL <https://doi.org/10.1201/9781351030144>. [p96]
- V. Amrhein and S. Greenland. Remove, rather than redefine, statistical significance. *Nature Human Behaviour*, 2(1):4, 2018. URL <https://doi.org/10.1038/s41562-017-0224-0>. [p97]
- V. Amrhein, S. Greenland, and B. McShane. Scientists rise up against statistical significance. *Nature*, 567(7748):305, 2019. URL <https://doi.org/10.1038/d41586-019-00857-9>. [p97]
- J. L. Arbuckle. *Amos*. IBM SPSS, Chicago, IL, USA, 2020. Version 27.0. [p99]
- BEsmarter Team. *BEsmarter*, 2020a. URL <https://besmarter-team.shinyapps.io/BEsmarter-GUI/>. Web application (shiny app). Accessed on October 15, 2020. [p99]
- BEsmarter Team. *BEsmarter source code*, 2020b. URL <https://github.com/besmarter/BSTApp>. Accessed on April 21, 2020. [p99]
- M. Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv:1701.02434v2 [stat]*, 2018. URL <https://arxiv.org/abs/1701.02434v2>. Accessed on March 7, 2021. [p98, 103]
- P.-C. Bürkner. brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1):1–28, 2017. URL <https://doi.org/10.18637/jss.v080.i01>. [p97, 100, 101]
- P.-C. Bürkner. Advanced Bayesian multilevel modeling with the R package brms. *The R Journal*, 10(1):395–411, 2018. URL <https://doi.org/10.32614/RJ-2018-017>. [p97, 100, 101]
- B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1):1–32, 2017. URL <https://doi.org/10.18637/jss.v076.i01>. [p97]
- W. Chang, J. Cheng, J. J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. *shiny: Web Application Framework for R*, 2022. URL <https://CRAN.R-project.org/package=shiny>. R package, version 1.7.2. [p97]
- S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987. URL [https://doi.org/10.1016/0370-2693\(87\)91197-X](https://doi.org/10.1016/0370-2693(87)91197-X). [p98]
- G. Emvalomatis. *BayES: Bayesian Econometrics Software*, 2020. URL <https://bayeconsoft.com/>. Version 2.5. Accessed on November 24, 2020. [p99]
- J. Gabry. *shinystan: Interactive Visual and Numerical Diagnostics and Posterior Analysis for Bayesian Models*, 2022. URL <https://mc-stan.org/shinystan/>. R package, version 2.6.0. [p97, 104]
- 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, 2019. URL <https://doi.org/10.1111/rssa.12378>. [p98]
- J. Gabry and R. Češnovar. *cmdstanr: R Interface to 'CmdStan'*, 2022. URL <https://mc-stan.org/cmdstanr>. R package, version 0.5.2. [p97]
- A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990. URL <https://doi.org/10.1080/01621459.1990.10476213>. [p98]
- A. Gelman. Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian Analysis*, 1(3):515–534, 2006. URL <https://doi.org/10.1214/06-BA117A>. [p96]
- A. Gelman, A. Jakulin, M. G. Pittau, and Y.-S. Su. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 2(4):1360–1383, 2008. URL <https://doi.org/10.1214/08-A0AS191>. [p96]
- A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, Boca Raton, FL, USA, 3rd edition, 2014. ISBN 978-1-4398-4095-5. URL <https://doi.org/10.1201/b16018>. [p96, 98]

- A. Gelman, D. Simpson, and M. Betancourt. The prior can often only be understood in the context of the likelihood. *Entropy*, 19(10):555, 2017. URL <https://doi.org/10.3390/e19100555>. [p96]
- A. Gelman, J. Hill, and A. Vehtari. *Regression and Other Stories*. Analytical Methods for Social Research. Cambridge University Press, Cambridge, UK, 2020a. ISBN 978-1-107-67651-0. URL <https://doi.org/10.1017/9781139161879>. [p96, 97]
- A. Gelman, A. Vehtari, D. Simpson, C. C. Margossian, B. Carpenter, Y. Yao, L. Kennedy, J. Gabry, P.-C. Bürkner, and M. Modrák. Bayesian workflow. *arXiv:2011.01808v1 [stat]*, 2020b. URL <https://arxiv.org/abs/2011.01808v1>. Accessed on March 7, 2021. [p97]
- S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984. URL <https://doi.org/10.1109/TPAMI.1984.4767596>. [p98]
- B. Goodrich, J. Gabry, I. Ali, and S. Brilleman. *rstanarm: Bayesian applied regression modeling via Stan*, 2022. URL <https://mc-stan.org/rstanarm>. R package, version 2.21.3. [p101]
- W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. URL <https://doi.org/10.1093/biomet/57.1.97>. [p98]
- M. D. Hoffman and A. Gelman. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(47):1593–1623, 2014. URL <https://jmlr.org/papers/v15/hoffman14a.html>. [p98]
- IBM Corp. *IBM SPSS Statistics for Windows*. IBM Corp., Armonk, NY, USA, 2020. Version 27.0. [p99]
- JASP Team. *JASP*, 2022. URL <https://jasp-stats.org/>. Version 0.16.3. [p99]
- A. A. Johnson, M. Q. Ott, and M. Dogucu. *Bayes Rules!: An Introduction to Applied Bayesian Modeling*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, New York, NY, USA, 2022. ISBN 978-0-429-28834-0. URL <https://doi.org/10.1201/9780429288340>. [p96]
- G. Karabatsos. A menu-driven software package for Bayesian regression analysis. *The ISBA Bulletin*, 22(4):13–16, 2015. [p96, 99]
- G. Karabatsos. A menu-driven software package of Bayesian nonparametric (and parametric) mixed models for regression analysis and density estimation. *Behavior Research Methods*, 49(1):335–362, 2017. URL <https://doi.org/10.3758/s13428-016-0711-7>. [p99]
- D. Lewandowski, D. Kurowicka, and H. Joe. Generating random correlation matrices based on vines and extended onion method. *Journal of Multivariate Analysis*, 100(9):1989–2001, 2009. URL <https://doi.org/10.1016/j.jmva.2009.04.008>. [p102]
- D. Lunn, D. Spiegelhalter, A. Thomas, and N. Best. The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, 28(25):3049–3067, 2009. URL <https://doi.org/10.1002/sim.3680>. [p98]
- D. J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter. WinBUGS - A Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10(4):325–337, 2000. URL <https://doi.org/10.1023/A:1008929526011>. [p99]
- D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2003. [p98]
- Y. Marchenko and N. Balov. In the spotlight: Bayesian “random-effects” models, 2015. URL <https://www.stata.com/stata-news/news30-2/bayesian-random-effects/>. Accessed on July 16, 2020. [p101]
- R. McElreath. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, Boca Raton, FL, USA, 2nd edition, 2020. ISBN 978-0-367-13991-9. URL <https://doi.org/10.1201/9780429029608>. [p96]
- B. B. McShane, D. Gal, A. Gelman, C. Robert, and J. L. Tackett. Abandon statistical significance. *The American Statistician*, 73:235–245, 2019. URL <https://doi.org/10.1080/00031305.2018.1527253>. [p97]
- N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. URL <https://doi.org/10.1063/1.1699114>. [p98]

- R. M. Neal. Probabilistic Inference using Markov Chain Monte Carlo Methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993. URL <https://www.cs.toronto.edu/~radford/review.abstract.html>. Accessed on March 21, 2022. [p98]
- R. M. Neal. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, Handbooks of Modern Statistical Methods. CRC Press, Boca Raton, FL, USA, 2011. ISBN 978-0-429-13850-8. URL <https://doi.org/10.1201/b10905>. [p98]
- J. Piironen, M. Paasiniemi, A. Catalina, F. Weber, and A. Vehtari. *projpred: Projection Predictive Feature Selection*, 2022. URL <https://mc-stan.org/projpred/>. R package, version 2.1.2. [p109]
- H. Qian. *Toolkit on Econometrics and Economics Teaching*. MATLAB Central File Exchange, 2011. URL <https://www.mathworks.com/matlabcentral/fileexchange/32601-toolkit-on-econometrics-and-economics-teaching>. MATLAB package, version from August 19, 2011. Accessed on March 26, 2020. [p99]
- A. Ramírez-Hassan and M. Graciano-Londoño. A GUIDed tour of Bayesian regression. *The R Journal*, 13(2):135–152, 2021. URL <https://doi.org/10.32614/RJ-2021-081>. [p97, 99]
- B. J. Reich and S. K. Ghosh. *Bayesian Statistical Methods*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, New York, NY, USA, 2019. ISBN 978-0-429-20229-2. URL <https://doi.org/10.1201/9780429202292>. [p96]
- rollApp, Inc. rollApp - Run desktop applications online, 2020. URL <https://www.rollapp.com/>. Accessed on April 24, 2021. [p109]
- rollApp, Inc. and JASP Team. JASP on rollApp, 2020. URL <https://www.rollapp.com/app/jasp>. Accessed on June 29, 2020. [p109]
- D. Spiegelhalter, A. Thomas, N. Best, and D. Lunn. *OpenBUGS User Manual, Version 3.2.3*, 2014. URL <http://www.openbugs.net/Manuals/Manual.html>. Accessed on March 25, 2020. [p99]
- Stan Development Team. *Stan Reference Manual, Version 2.29*, 2022a. URL https://mc-stan.org/docs/2_29/reference-manual/index.html. Accessed on April 13, 2022. [p98]
- Stan Development Team. *RStan: The R Interface to Stan*, 2022b. URL <https://mc-stan.org/>. R package, version 2.21.5; for the example, version 2.26.13 from the repository at <https://mc-stan.org/r-packages/> was used. [p97]
- Stan Development Team. Runtime warnings and convergence problems, 2022c. URL <https://mc-stan.org/misc/warnings.html>. Version from March 10, 2022. Accessed on April 13, 2022. [p103]
- Stan Development Team. *Stan Modeling Language Users Guide and Reference Manual, Version 2.29*, 2022d. URL <https://mc-stan.org>. [p97]
- StataCorp. Introduction to Bayesian analysis. In *Stata Bayesian Analysis Reference Manual*. Stata Press, College Station, TX, USA, 2019a. ISBN 978-1-59718-272-0. URL <https://www.stata.com/manuals/bayesintro.pdf>. Release 16. Accessed on November 11, 2020. [p96]
- StataCorp. *Stata*. StataCorp LLC, College Station, TX, USA, 2019b. Release 16. [p99]
- A. Van Welzen, M. Hoch, P. Wahl, F. Weber, S. Rode, J. K. Tietze, L. Boeckmann, S. Emmert, and A. Thiem. The response and tolerability of a novel cold atmospheric plasma wound dressing for the healing of split skin graft donor sites: A controlled pilot study. *Skin Pharmacology and Physiology*, 34(6):328–336, 2021. URL <https://doi.org/10.1159/000517524>. [p104]
- A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner. Rank-normalization, folding, and localization: An improved \hat{R} for assessing convergence of MCMC (with discussion). *Bayesian Analysis*, 16(2):667–718, 2021. URL <https://doi.org/10.1214/20-BA1221>. [p103]
- F. Weber. *shinybrms: Graphical User Interface ('shiny' App) for 'brms'*, 2022. URL <https://fweber144.github.io/shinybrms/>. R package, version 1.8.0. [p97, 100]
- F. Weber, G. Knapp, Ä. Glass, G. Kundt, and K. Ickstadt. Interval estimation of the overall treatment effect in random-effects meta-analyses: Recommendations from a simulation study comparing frequentist, Bayesian, and bootstrap methods. *Research Synthesis Methods*, 12(3):291–315, 2021. URL <https://doi.org/10.1002/jrsm.1471>. [p97]
- P. Woodward. BugsXLA: Bayes for the common man. *Journal of Statistical Software*, 14(1):1–18, 2005. URL <https://doi.org/10.18637/jss.v014.i05>. [p97, 98]

P. Woodward. *Bayesian Analysis Made Simple: An Excel GUI for WinBUGS*. Chapman & Hall/CRC Biostatistics Series. CRC Press, Boca Raton, FL, USA, 2011. ISBN 978-1-4398-3954-6. URL <https://doi.org/10.1201/b11235>. [p99]

Frank Weber

*Institute for Biostatistics and Informatics in Medicine and Ageing Research
Rostock University Medical Center
Ernst-Heydemann-Str. 8
18057 Rostock
Germany
ORCID: 0000-0002-4842-7922
frank.weber@uni-rostock.de*

Katja Ickstadt

*Department of Statistics
TU Dortmund University
Vogelpothsweg 87
44227 Dortmund
Germany
ORCID: 0000-0001-5157-2496
ickstadt@statistik.tu-dortmund.de*

Anne Glass

*Institute for Biostatistics and Informatics in Medicine and Ageing Research
Rostock University Medical Center
Ernst-Heydemann-Str. 8
18057 Rostock
Germany
ORCID: 0000-0002-7715-9058
aenne.glass@uni-rostock.de*

Quantifying Population Movement Using a Novel Implementation of Digital Image Correlation in the ICvectorfields Package

by Devin W. Goodsman

Abstract Movements in imagery captivate the human eye and imagination. They are also of interest in variety of scientific disciplines that study spatiotemporal dynamics. Popular methods for quantifying movement in imagery include particle image velocimetry and digital image correlation. Both methods are widely applied in engineering and materials science, but less applied in other disciplines. This paper describes an implementation of a basic digital image correlation algorithm in R as well as an extension designed to quantify persistent movement velocities in sequences of three or more images. Algorithms are applied in the novel arena of landscape ecology to quantify population movement and to produce vector fields for easy visualization of complex movement patterns across space. Functions to facilitate analyses are available in **ICvectorfields** (Goodsman, 2021). These methods and functions are likely to produce novel insights in theoretical and landscape ecology because they facilitate visualization and comparison of theoretical and observed data in complex and heterogeneous environments.

1 Introduction

Living organisms move through space in complex ways that have inspired many branches of spatial pattern analysis from Turing instabilities (Alonso, Bartumeus, and Catalan 2002; Ruan 1998), to complex systems analysis of the emergent properties of individual-level behaviours when organisms live in groups (Parrish and Edelstein-Keshet 1999; N. Johnson 2009). Moreover, in mathematical ecology there is a long history of deriving analytic expressions for traveling wave speeds from mathematical models of biological systems based on partial differential equations Skellam (1951) and integrodifference equations (Kot, Lewis, and Driessche 1996). In addition to standard traveling waves and wave-trains, simulation studies have revealed more unusual patterns of population level movement can arise from the way organisms interact and move on the landscape (Hassell, Comins, and May 1994). Spiral waves are one example of surprising spatiotemporal dynamics that can arise in biological systems (Hassell, Comins, and May 1991).

Travelling waves and spiral waves emerge from mathematical models of population expansion, which are often based on partial differential equations and integrodifference equations. These types of models, which represent the movement patterns of populations of organisms, are sometimes classified as Eulerian approaches to distinguish them from Lagrangian approaches that focus on the trajectories of individuals. The majority of R packages that quantify organismal movement, however, are Lagrangian as they pertain to the analysis of the tracks or trajectories of individual animals with tracking collars or tracking devices. Integrated step selection models (Avgar et al. 2016), such as those in the **amt** package (Signer, Fieberg, and Avgar 2019), which incorporate the impact of spatially variable habitat or environmental variables on movement of individuals modeled using a discrete time and discrete space framework, are an example of a Lagrangian approach when fitted to movement data from individuals. Because my focus in this work is on population-level movements that are evident in imagery, I will forego further discussion of Lagrangian models and instead refer the interested reader to a review of R packages for modeling animal movement (Joo et al. 2020). At the time of writing, R packages that focus on the Eulerian approach include **IDE** (Zammit-Mangion 2019), **deSolve** (Soetaert, Petzoldt, and Setzer 2010), and **ReacTran** (Soetaert and Meysman 2012). These packages are designed primarily to obtain numerical solutions to Eulerian models, analyze their dynamics, and fit them to data. Recently, movement modeling based on stochastic differential equations, stochastic partial differential equations (Krainski et al. 2018), and other stochastic process models (Buderman et al. 2016) has proliferated. Computationally efficient Bayesian statistical approaches are often required to fit these stochastic models to data due to the ubiquity of noise in spatiotemporal time series in combination with nonlinear dynamical processes (Krainski et al. 2018).

In contrast to the R packages and approaches I have cited above, this work is focused on the description of an empirical method for quantifying spatially heterogeneous rates of movement in sequences of images without fitting a model—although I do approach the quantification problem from an Eulerian perspective. Empirically quantifying spread rates without imposing a specific mathematical model allows the user to abandon many of the assumptions implicit in mathematical modeling of population expansion. For example, tractable mathematical models of consumer-resource

systems that generate traveling waves, wave-trains, and spiral waves, often rely on assumptions of a homogeneous spatial environment with respect to resources or one in which there are no discontinuities in resources. In contrast, many organisms spread in environments that are spatially heterogeneous (Urban et al. 2008), and in environments subject to persistent directional flows that impact organism movement (Hoffman et al. 2006). The ramifications of this claim are more easily understood using a meteorological analogy. In meteorology, vector fields are frequently used to illustrate the impacts of high and low pressure systems on wind speed and direction, and thus on the movement of weather systems. In such meteorological systems, wind speed and direction are complicated functions of topography and complex atmospheric dynamics. As a result, vector fields representing movement in such systems are often variable at the regional scale, with winds flowing in one direction on one side of a map and possibly in an opposing direction on the other side. In ecology, populations of organisms are like the weather systems in that their movement on the landscape is what is of primary interest to researchers; variable wind causes spatially variable movement of weather systems similarly to how persistent directional fluid flows, including wind, in an organism's environment impact dispersal, and therefore population movement.

At the time of writing, the only tool in R (R Core Team 2021) designed to empirically estimate spreading speed or the speed of wave-trains in populations without fitting a mathematical model is implemented in the `ncf` R package (Bjørnstad 2020). The `ncf` package relies on lagged non-parametric cross-correlation functions to estimate spreading speed of traveling waves (Bjørnstad and Falck 2001). To do so, it takes two spatiotemporal data sets that differ from one another only in that one is a time-lagged version of the other, and projects their planar coordinates onto lines of varying angles that can be specified using function arguments. After projection onto a line, cross-correlation is estimated using a spline-correlogram approach (Bjørnstad and Falck 2001) and the location of maximum cross-correlation gives an estimate of displacement along the direction of the projection line. This approach was used to estimate the velocity of traveling wave-trains in the larch budmoth system in the European Alps (Bjørnstad et al. 2002).

Projecting population data from a domain with two spatial coordinates onto a domain that has only one spatial coordinate and then using a correlogram approach precludes quantification of more complicated patterns of movement on the landscape. For example if two spatially separated populations are moving towards one another at the same speed, such an approach will yield enigmatic correlograms. Similarly, if several populations move radially around a central fulcrum, the correlogram will be difficult to translate to an estimate of directional movement.

In this paper I present an approach for estimating vector-fields in systems with spatially variable movement that is inspired by a technique from engineering and materials science called Digital Image Correlation or DIC (Anuta 1970; Sutton, Orteu, and Schreier 2009). Among other things, Digital Image Correlation is used to estimate displacement based on photographs of a planar material before, during, and after a force has been applied to warp its surface (Sutton, Orteu, and Schreier 2009). A typical DIC approach as well as the extensions described in this paper are implemented in the `ICvectorfields` package (Goodsman 2021), in which the IC is the abbreviation for Image Correlation. I demonstrate these approaches using the `ICvectorfields` package to analyze a simulated data set as well as the larch budmoth data set provided with the `ncf` R package (Bjørnstad 2020; Bjørnstad et al. 2002).

2 Mathematical and Computational Details

Here I provide mathematical and computational details of the algorithms used in the `ICvectorfields` R package starting with a standard digital image correlation approach, and following with extensions to estimate persistent movement and to account for spatial variability in persistent movement. The `ICvectorfields` package capitalizes on the algorithms written in C under the title FFTW which stands for Fastest Fourier Transform in the West (Frigo and Johnson, 2005), and a convenient wrapper package in R called `fftwtools` (Rahim 2021). Input raster images and raster stacks are read and manipulated using the `terra` package (Hijmans 2021).

Digital image correlation

One of the earliest applications of cross-correlation in image analysis was to align images taken from different sensors or at different times using satellites or aircraft (Anuta 1970). The theoretical and computational details I present here loosely follow those in this pioneering application. I will provide the mathematical underpinning of two-dimensional cross-correlation, and then elaborate on its computational implementation, which involves some additional complexity due to the circular nature of discrete fast Fourier transforms. In all descriptions below, I do not normalize the cross-correlation function to obtain Pearson correlation coefficients and therefore, I follow the convention of using the

terms cross-correlation and cross-covariance interchangeably.

Given two images that have been converted to square matrices f and g of dimension $m \times m$, two-dimensional cross-correlation can be defined in terms of a convolution:

$$(f \star g)(x_j, y_i) = \left(\overline{f(-x_j, -y_i)} * g(x_j, y_i) \right)(x_j, y_i), \tag{1}$$

in which $(f \star g)$ is the two-dimensional cross-correlation matrix, the $*$ operator denotes convolution, $\overline{f(-x_j, -y_i)}$ is the complex conjugate of the $f(x_j, y_i)$ matrix, i is the matrix row index, and j is the matrix column index $i, j \in \mathbb{N} = \{1, 2, \dots\}$. Note that I use array indices that start at one rather than zero. The coordinates of the centroids of each pixel are given by x_j and y_i .

Based on the convolution theorem, equation (1) can be rewritten as

$$(f \star g)(x_j, y_i) = \mathbb{F}^{-1} \left(\overline{\mathbb{F}(f(x_j, y_i))} \mathbb{F}(g(x_j, y_i)) \right)(x_j, y_i), \tag{2}$$

wherein \mathbb{F} denotes the two-dimensional Fourier transform, \mathbb{F}^{-1} denotes its inverse, and $\overline{\mathbb{F}(f(x_j, y_i))}$ is the complex conjugate of $\mathbb{F}(f(x_j, y_i))$. Because $\overline{\mathbb{F}(f(x_j, y_i))} = \mathbb{F}(f(-x_j, -y_i))$, and because $f(x_j, y_i)$ contains only real numbers, the complex conjugate can be calculated using matrix multiplication:

$$\overline{f(-x_j, -y_i)} = r \times f \times r, \tag{3}$$

in which the r matrix is a $m \times m$ matrix that has zeros everywhere except for along the diagonal that runs from its lower left to upper right corners, which contains ones:

$$r = \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{pmatrix}. \tag{4}$$

The matrix calculations in equations (3) and (4) substitute for calculation of the complex conjugate only in the case where all values of the f matrix are real, which is the case in most natural science applications.

Together, equations (1) through (4) constitute an elegant way to compute two dimensional cross-correlation. Computer implementation of these, however, requires some additional complexity due to the use of discrete fast Fourier transforms to efficiently compute convolutions. Discrete fast Fourier transforms are inherently circular, which means that what happens on the outer edges of matrices will impact their discrete fast Fourier transform on the opposite side. In order to mitigate this problem, zeros are added to the outer edge on all sides of both the f and g matrices (Anuta 1970). In **ICvectorfields**, the f and g matrices are padded with as many zeros as there are rows and columns in the original matrix and then additional zeros are added to ensure that both matrices are square with an even number of rows and columns.

When matrices are padded as described above and discrete fast Fourier transforms are used as in equation (2), the non-cyclic components of the convolution end up in the outer corners of the cross-correlation matrix $(f \star g)(x_j, y_i)$ (Anuta 1970). Thus, to obtain a correct estimate of cross-correlation, $(f \star g)(x_j, y_i)$ must be divided into four quadrants and each quadrant must be flipped along its horizontal and vertical axes using matrix multiplication. For example, if the zero-padded matrices have dimension $k \times k$, each quadrant of $(f \star g)(x_j, y_i)$ will have dimension $k/2 \times k/2$ and the following operation flips each quadrant matrix:

$$q_f = r \times q \times r, \tag{5}$$

where r is a $k/2 \times k/2$ matrix as in equation (4). Then the four quadrants can be reassembled into the $k \times k$ cross-correlation matrix. The mathematical operations in equations (1) through (5) are implemented in the `Xcov2D` function in the **ICvectorfields** package.

Once a cross-correlation matrix has been estimated, it can be used to compute displacement in the

horizontal and vertical directions in terms of the horizontal and vertical shifts in pixel numbers that maximize cross-correlation. In **ICvectorfields**, shifts to the right and up are designated as positive, whereas shifts to the left and down are considered negative.

A typical implementation of DIC will define a region of interest within the input images or their corresponding matrices wherein displacement vectors are sought using a bounding box as in the *DispFieldbb* function in **ICvectorfields** or using a sub-grid of equal sized regions of interest as in the *DispField* function in **ICvectorfields**. Note that all of the functions in **ICvectorfields** that use DIC or variations of it, translate displacement or velocities in terms of pixel shifts to the spatial units defined in the projection information of the original input rasters. The coordinate information required for translation of pixel shifts to the correct spatial units is obtained using functions in the **terra** R package (Hijmans 2021).

Extending DIC to quantify persistent movement

In applications of DIC in earth systems with persistent directional flows that influence movement, it is valuable to determine directional movement of populations of interest that persist for more than one time step. In such situations, a spatiotemporal array of images with two space dimensions and one time dimension is required. Often these can be formulated as stacks of raster images, with each layer in the stack representing spatially referenced observations for one time step (step one in Fig. 1). A variation of DIC which I call Spatiotemporal Image Correlation (STIC) permits estimation of persistent directional movement in terms of orthogonal velocity vectors.

In STIC, the three dimensional array is first lagged by duplicating it and then removing an integer number of layers from the top of one duplicate and the bottom of the other (steps two and three in Fig. 1). The integer lag is user defined and serves to minimize estimates of zero movement which always would occur in the absence of a lag. To differentiate the duplicate lagged arrays, I will refer to the first as the reference array, and the second as the lagged array. Regions of interest in the reference array are selected and locations outside the region of interest in the reference array are assigned values of zero (grey shaded region in steps two and three in Fig. 1 represent regions of interest). The reference array and the lagged array are then dimension reduced by averaging along rows to obtain one pair of two-dimensional matrices and by averaging along columns to obtain a second pair of two dimensional matrices (step 4 in Fig. 1). The first pair of matrices comprises row-averaged reference and lagged matrices. The second pair of matrices comprises column-averaged reference and lagged matrices. Each matrix in the two pairs has one space dimension and one time dimension.

Cross-correlation between the pairs of reference and lagged matrices is then computed as described for DIC. Recall that one dimension of each of the row or column-averaged matrices is spatial while the other is temporal, which enables calculation of two orthogonal velocity vectors based on space shifts and time shifts obtained by application of DIC:

$$v_x = s_x / s_{tx}, s_{tx} \neq 0 \quad (6)$$

$$v_y = s_y / s_{ty}, s_{ty} \neq 0 \quad (7)$$

in which v_x and v_y are velocity in the horizontal and vertical directions, s_x and s_y are shifts in the horizontal and vertical direction, s_{tx} is the time shift that corresponds to spatial shifts in the horizontal direction, and s_{ty} is the time shift that corresponds to spatial shifts in the vertical direction. Note that due to the time shift, the user-defined time lag does not necessarily pre-determine the denominator of the orthogonal velocity vectors.

Spatially variable velocities

When the magnitudes of movement velocities are highly spatially variable, a single time lag is not optimal for quantifying orthogonal velocity vectors. For these scenarios a variation on the STIC algorithm called STIC+ allows the user to specify a maximum time lag. The algorithm then repeats the steps described for STIC for each integer time lag from one to the maximum time lag. For each repetition and each location of interest, the total velocity magnitude (speed) is calculated as

$$|v| = \sqrt{v_x^2 + v_y^2} \quad (8)$$

For each region of interest, the horizontal and vertical velocity vectors are determined by the time lag STIC calculation that maximizes equation (8).

A summary table describing which functions in **ICvectorfields** use each of the algorithms described

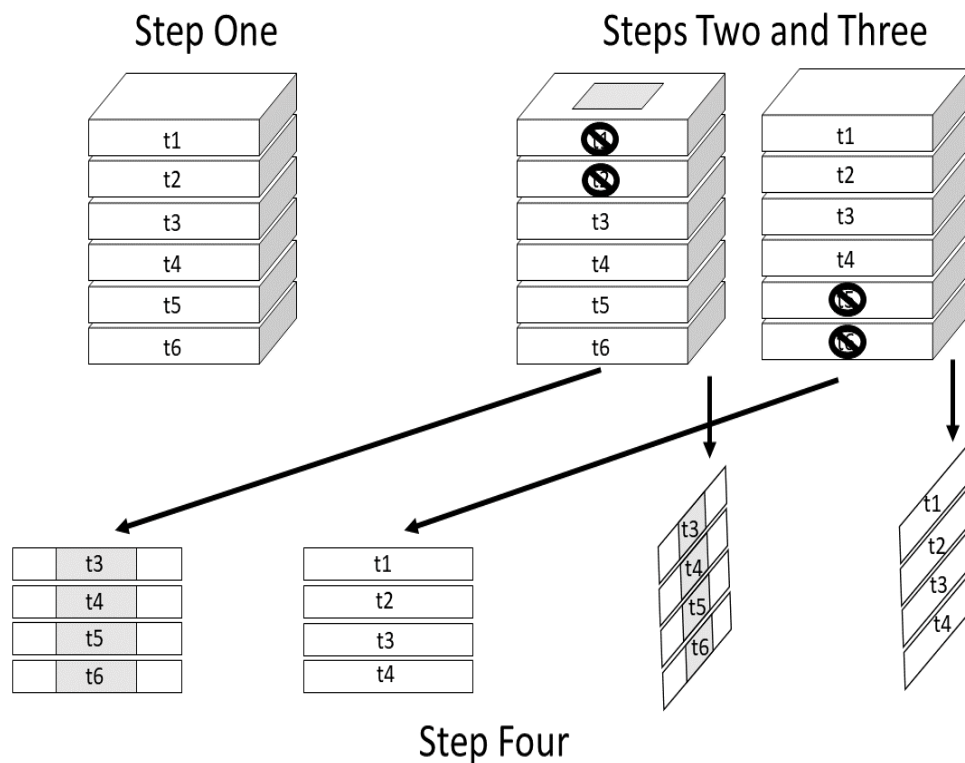


Figure 1: The STIC algorithm: The input array in step one is a raster stack of images in which each image layer represents the phenomenon of interest in planar space at a different time instance. In step two, the input array is duplicated and based on a user specified time lag, layers are removed from the top of one array and the bottom of the other. In addition, a region of interest is defined in one of the duplicate arrays represented by the grey shaded region at the top of the prism on the left. In step four the rows are dimension reduced by averaging along one of the axes (either rows or columns). This produces a pair of row-averaged matrices and a pair of column-averaged matrices that are analyzed using cross-correlation to estimate orthogonal velocity vectors.

Table 1: ICvectorfields functions, algorithms, and use contexts to facilitate decisions on which function is most applicable. ROI stands for region of interest, which is defined either using a grid or a bounding box, Velocities refers to whether the magnitudes of velocities in the vector field are presumed to be spatially variable or not.

function	algorithm	images	ROI	velocities
DispField	DIC	2	grid	variable or not
DispFieldbb	DIC	2	bounding box	variable or not
DispFieldST	STIC	3+	grid	less variable
DispFieldSTbb	STIC	3+	bounding box	less variable
DispFieldSTall	STIC+	3+	grid	more variable
DispFieldSTball	STIC+	3+	bounding box	more variable

above is provided (Table 1). Two functions in **ICvectorfields** use a standard implementation of DIC similar to that described by (Anuta 1970), two functions use the STIC extension and two functions use the STIC+ extension (Table 1).

3 Application

In this section I demonstrate the use of DIC and extensions implemented in **ICvectorfields** (Goodsman 2021) using an example in which data were simulated based on a partial differential equation and using the classic larch budmoth defoliation data that are embedded in the **ncf** package (Bjornstad 2020). The data from the simulated example are embedded in **ICvectorfields**. For visualization of results, the demonstrations call functions in several R packages: These include **ggplot2** (Wickham 2016), **ggnewscale** (Campitelli 2021a), **metR** (Campitelli 2021b), and **terra** (Hijmans 2021).

Demonstration using simulated data

The model used to simulate data to test and demonstrate the functionality of **ICvectorfields** is a convection-diffusion equation, which is a partial differential equation with terms for diffusion, advection, and reaction:

$$\frac{\partial u}{\partial t} = \nabla \cdot (D \nabla u) - \nabla \cdot (\mathbf{v}u) + ru, \quad (9)$$

in which r is the growth rate with units of per unit time, D is the diffusion coefficient with squared spatial units per unit time, \mathbf{v} is the bivariate orthogonal velocity vector in units of space per unit time, ∇ represents the gradient, and $\nabla \cdot$ represents divergence. The orthogonal velocity vector is spatially dependent in the simulations that follow:

$$\mathbf{v} = \begin{cases} (0, 0.2), & x < 0, y \geq 0 \\ (0.2, 0), & x \geq 0, y \geq 0 \\ (0, -0.2), & x \geq 0, y < 0 \\ (-0.2, 0), & x < 0, y < 0 \end{cases} \quad (10)$$

Note that by convention in equations (9) and (10), movement to the right and up has a negative sign, and movement to the left and down has a positive sign. This is the opposite convention used in **ICvectorfields**. Note that the discontinuities at $x = 0$ and $y = 0$ in the advection term in equation (10) create strange model behaviours once concentration reaches $x = 0$ or $y = 0$, and so the model was simulated for only 6 time steps to minimize encounters with these axes. Parameter values for the diffusion coefficient and the growth rate were $D = (0.01, 0.01)$ squared spatial units per unit time and $r = 0.5$ per unit time.

The model in equations (9) and (10) was simulated using the **ReacTran** R package (Soetaert and Meysman 2012), using a finite differencing scheme with backward differencing on a square domain of 202 cells in each direction, each with a width of 0.049 spatial units. The initial condition was a concentration of one units per arbitrary unit of volume in the centre of each quadrant of the spatial domain. Boundary conditions were zero flux (reflecting) on all four sides of the spatial domain. The simulation data are saved in table format within **ICvectorfields**.

The data are imported and then converted from table format to a raster stack using the **RastStackData** helper function. They can then be visualized as rasters as shown in Fig. 2.

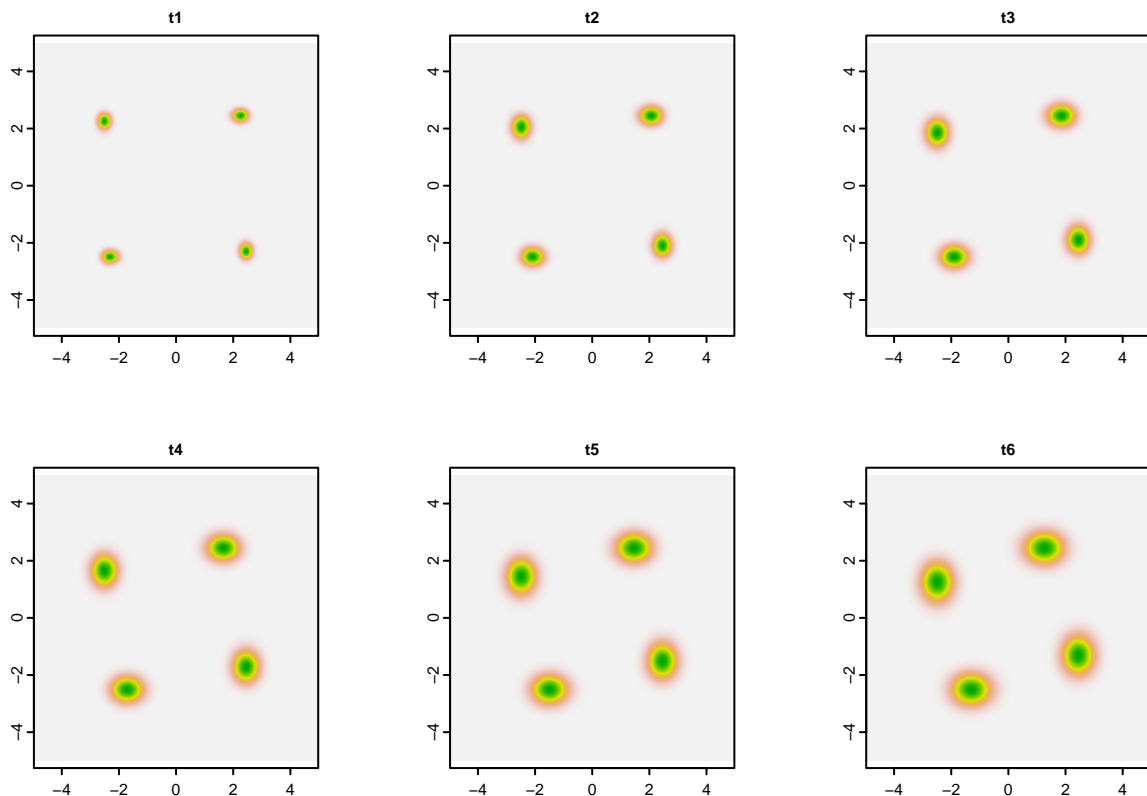


Figure 2: Visualization of simulation data for six time steps. The initial condition at t_0 is not shown. Green colours represent the highest concentrations. The populations in each quadrant move counter-clockwise as time steps forward.

```
# import simulated data
data(SimData, package = "ICvectorfields")

# convert to raster stack
SimStack <- ICvectorfields::RastStackData(SimData)

# confirming dimension
#dim(SimStack)

# plotting
layout(matrix(1:6, 2, 3, byrow = TRUE))
#layout.show(6)
terra::plot(SimStack[[1]], legend = FALSE, main = "t1")
terra::plot(SimStack[[2]], legend = FALSE, main = "t2")
terra::plot(SimStack[[3]], legend = FALSE, main = "t3")
terra::plot(SimStack[[4]], legend = FALSE, main = "t4")
terra::plot(SimStack[[5]], legend = FALSE, main = "t5")
terra::plot(SimStack[[6]], legend = FALSE, main = "t6")
```

To analyze displacement based on a pair of images, I chose to use the standard implementation of DIC in the *DispField* function of *ICvectorfields*. The first two arguments of the *DispField* function are the input rasters. The first input raster is treated as the reference image and the second is treated as the shifted image. In this case, the first image is the raster layer corresponding to the first time step (t_1) and the second image is the raster layer corresponding to the sixth time step (t_6). The function selects regions of interests based on a grid of dimensions given in the *factv1* and *facth1* arguments, which represent to number of rows and columns in each sub-grid. Sub-grids start in the upper left corner and as many sub-grids as fit within the original domain are constructed. In the code below, sub-grids are 101×101 , which is approximately the size of one quarter of the original spatial domain of the simulation. The *restricted* argument is by default set to *FALSE*. In that case, the DIC algorithm cross-correlates each region of interest in the first image with the entirety of the second image. When *restricted = TRUE*, the algorithm cross correlates both images only within the region of interest. If the user has reason to believe that movement is predominantly occurring within sub-grids the *restricted =*

Table 2: ICvectorfields output from a call of the `DispField` function using simulated data. The table is a duplicate of the data table returned after calling the function except that columns 3 through 6 have been omitted so that the table fits within page width limits.

rowcent	colcent	centx	centy	dispx	dispy
51	51	-2.487624	2.487624	0.0000000	-0.9851975
152	51	-2.487624	-2.487624	0.9851975	0.0000000
51	152	2.487624	2.487624	-0.9851975	0.0000000
152	152	2.487624	-2.487624	0.0000000	0.9851975

`TRUE` option has the added benefit of speeding up computation.

```
# Estimating displacement of simulated data using the DispField function
VFdf1 <- DispField(SimStack[[1]], SimStack[[6]], factv1 = 101, facth1 = 101,
  restricted = TRUE)
```

The output of `DispField` is in data table format. Because the data table is small, the output is duplicated in Table 2.

The directions of movement coincide with the directions of advection in the simulation with movement downwards in the upper left quadrant (first row of Table 2), movement to the right in the lower left quadrant (second row of Table 2), movement to the left in the upper right quadrant (third row of Table 2), and upwards movement in the lower right quadrant (fourth row of Table 2). Speed of movement can be computed by dividing displacement by the number of time steps that passed $0.98/5 = 0.196$, which is slightly slower than the simulated advection speed of 0.2 spatial units per time step. The discrepancy is likely due to the blurring effect of diffusion in the partial differential equation (equation (9)).

In situations where the speed is constant, velocity can be estimated from pairs of images as I have demonstrated above. However, the `DispFieldST` algorithm is designed to return orthogonal velocity vectors and so for confirmation purposes, I demonstrate it below:

```
# Estimating orthogonal velocity vectors of simulated data using the DispFieldST function
VFdf2 <- DispFieldST(SimStack, lag1 = 1, factv1 = 101, facth1 = 101, restricted = TRUE)
```

The data table that is returned after running the code above looks similar to the data table duplicated in Table ?? except that under the heading `dispx` and `dispy` the algorithm returns horizontal and vertical velocities rather than displacement vectors. The directions of movement are the same as those shown in Table 2, but the speed is 0.196 spatial units per unit time as previously estimated.

The vector field and the raw data can be visualized simultaneously using plotting functionality in `ggplot2` with extensions in the `ggnewscale` and `metR` packages.

```
SimVF <- ggplot() +
  xlim(c(-5, 5)) +
  ylim(c(-5, 5)) +
  geom_raster(data = SimData,
    aes(x = xcoord, y = ycoord, fill = t1)) +
  scale_fill_gradient(low = "white", high = "blue", na.value = NA) +
  new_scale("fill") +
  geom_raster(data = SimData,
    aes(x = xcoord, y = ycoord, fill = t6), alpha = 0.5) +
  scale_fill_gradient(low = "white", high = "red", na.value = NA) +
  geom_vector(data = VFdf2,
    aes(x = centx, y = centy,
      mag = Mag(dispx, dispy),
      angle = Angle(dispx, dispy))) +
  theme(panel.background = element_rect(fill = "white", color = "grey"),
    legend.key.size = unit(0.4, 'cm'))
```

SimVF

The resulting figure is duplicated in Fig. 3. The velocity vectors in the vector field are consistent with the simulated advection vectors (Fig. 3), although they slightly underestimate movement speed.

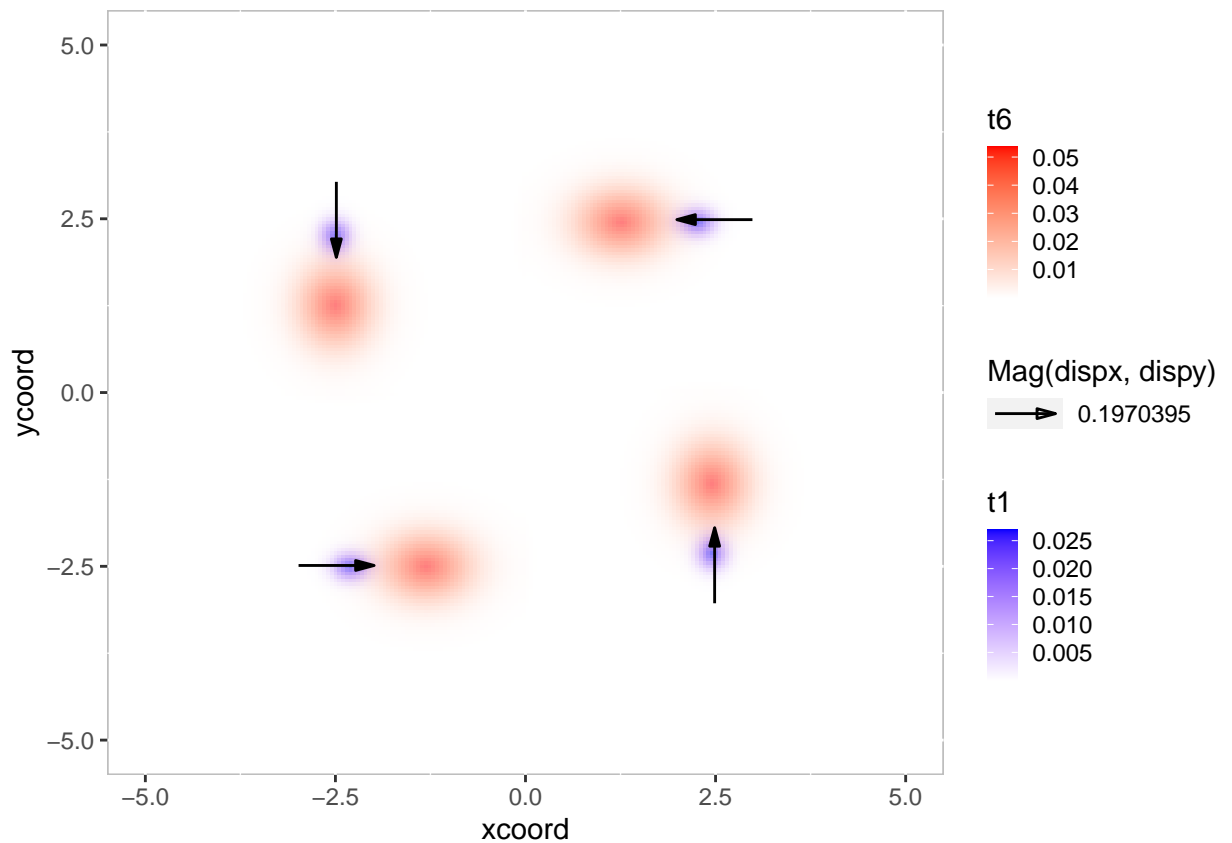


Figure 3: Vector field for radial movement simulated using a convection-diffusion equation. The orthogonal velocity vectors are estimated using the `DispFieldST` function in the `ICvectorfields` package.

Table 3: Output from a call of the `DispFieldST` function using simulated data. This call is meant to demonstrate a potential pitfall in using the cross-correlation approach because when `restricted = FALSE`, the algorithm finds positive cross-correlations that are not caused by movement. The table is a duplicate of the data table returned after calling the function except that columns 3 through 6 have been omitted so that the table fits within page width limits.

rowcent	colcent	centx	centy	dispx	dispy
51	51	-2.487624	2.487624	0.0985198	-3.8915302
152	51	-2.487624	-2.487624	3.8915302	0.0985198
51	152	2.487624	2.487624	-3.8915302	-0.0985198
152	152	2.487624	-2.487624	-0.0985198	3.8915302

Before proceeding to the next demonstration I will illustrate one of the potential pitfalls of estimating movement based on cross-correlation. If the argument of the `DispFieldST` function is left its default `restricted = FALSE` configuration, the algorithm will search the entire domain for shifts that maximize cross-correlation. Because the simulations in each quadrant of the spatial domain are quite similar, cross-correlation is in fact maximized by shifts that cross quadrants, even though simulated movement was not that large. Therefore, calling `DispFieldST` with `restricted = FALSE` produces incorrect output (Table 3): The simulated advection speed is not at all close to the estimated maximum orthogonal advection speed of 3.89 spatial units per unit time.

```
# Estimating orthogonal velocity vectors of simulated data using the DispFieldST function
VFdf3 <- DispFieldST(SimStack, lag1 = 1, factv1 = 101, facth1 = 101, restricted = FALSE)
```

Demonstration using larch budmoth data

Larch budmoths are lepidopteran defoliators that exhibit periodic outbreaks every 8 to 9 years in the European Alps (Bjørnstad et al. 2002). The larch budmoth data originated from survey information collected by the forest administrative agencies of France, Italy, Switzerland, and Austria from 1961 to

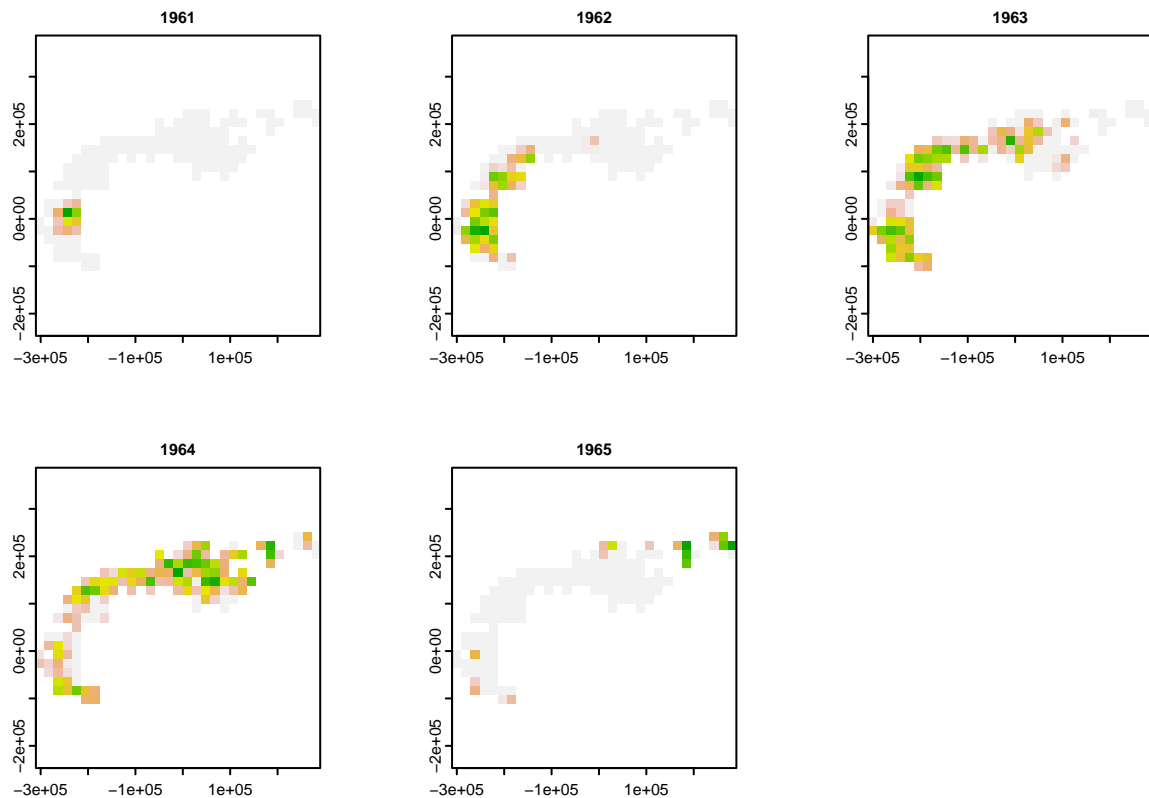


Figure 4: The first five years of the larch budmoth defoliation data included in the `ncf` package. Green colours represent the highest level of defoliation. Defoliation intensity moves from the southwest to the northeast but not along a straight trajectory between the southwest and northeast corners.

1998. The data record the presence of defoliation by larch budmoth caterpillars within 1×1 km grid cells (a binary variable). These data were aggregated up to 20×20 km grid cells so that records at this spatial scale were population proxies for larch budmoth caterpillar abundance (Bjørnstad et al. 2002) based on the assumption that defoliation damage is proportional to the abundance of the causal agents. Grid cells were excluded from the data set if they exhibited less than one percent defoliation or if more than ninety percent of years in which data were collected at that location exhibited no defoliation by larch budmoth (Bjørnstad et al. 2002). The larch budmoth defoliation data exhibit directional traveling wave-trains that travel from the southwest to the northeast along the European Alps (Bjørnstad et al. 2002). These data are embedded in the `ncf` R package.

After loading the `ncf` package as well as `ICvectorfields`, the data can be loaded, converted to a raster stack and visualized as follows:

```
# import larch budmoth data
data(lbm, package = "ncf")

# convert to raster stack
LBMStack <- ICvectorfields::RastStackData(lbm)

# confirming dimension
#dim(LBMStack)

# visualizing
layout(matrix(1:6, 2, 3, byrow = TRUE))
#layout.show(6)
terra::plot(LBMStack[[1]], legend = FALSE, main = "1961")
terra::plot(LBMStack[[2]], legend = FALSE, main = "1962")
terra::plot(LBMStack[[3]], legend = FALSE, main = "1963")
terra::plot(LBMStack[[4]], legend = FALSE, main = "1964")
terra::plot(LBMStack[[5]], legend = FALSE, main = "1965")
```

This code plots the first five years of the data set (Fig. 4), which show a standard progression

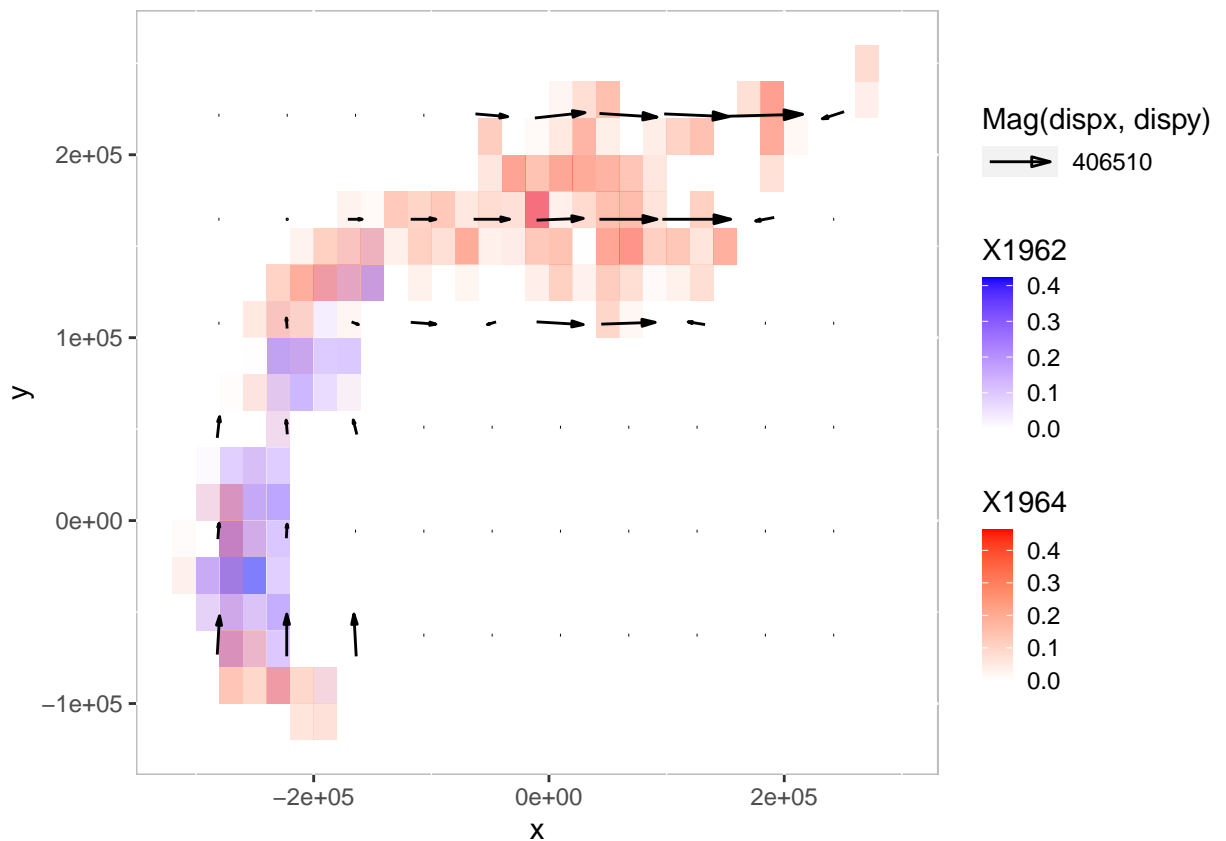


Figure 5: Vector field for Larch Budmoth persistent movement. The orthogonal velocity vectors are estimated using the `DispFieldSTall` function in the `ICvectorfields` package. Blue colours show the locations and intensities of defoliation in 1962 and red colours show the locations and intensities of defoliation in 1964. Vectors have their own scale that is distinct from the scale of the map. The figure shows that velocities point predominantly to the north on the southwest corner of the map and predominantly to the east on the northeast corner of the map. In addition, outbreaks appear to slow down as they turn the corner from southwest to northeast.

of outbreaks from the southwest corner of the Alps to the northeast. This pattern repeats relatively regularly every 8 to 10 years in the data set.

The study region covers a large geographic area and so it is likely the population movement speeds vary geographically. For this reason, I elected to use the STIC+ algorithm to analyze the data using `DispFieldSTall`. In the code below I analyze the first 23 years of the time series (1961 to 1983) as defoliation patterns from 1984 to 1998 are less regular.

```
# calculating velocity field for larch budmoth
VFdf3 <- DispFieldSTall(LBMStack[[1:23]], lagmax = 3, factv1 = 3, facth1 = 3, restricted = FALSE)
```

Calling `DispFieldSTall` returns a data frame object that is convenient for plotting the vector field. The vector field reveals that moth movement is to the north on the southwestern side of the Alps and then to the east on the northern side of the Alps (Fig. 5). It also reveals deceleration as outbreaks turn the corner and then acceleration as outbreaks move eastward (Fig. 5).

The average speed of larch budmoth movement can be computed from the data frame output of `DispFieldSTall` as follows:

```
# calculating average speed of population movement
VFdf3$speed <- sqrt((VFdf3$dispx^2) + VFdf3$dispy^2)

# sub-setting to remove locations where speed is zero
VFdf4 <- subset(VFdf3, speed > 0)

# computing mean, standard deviation and dimension of data frame
# to obtain sample size
mean(VFdf4$speed)
```

```

#> [1] 175810

#sd(VFdf4$speed)
#dim(VFdf4)

# upper and lower Wald-type 95 percent confidence interval on average speed
mean(VFdf4$speed)/1000 + qt(0.975, dim(VFdf4)[1] - 1)*sd(VFdf4$speed)/1000/sqrt(dim(VFdf4)[1] - 1)

#> [1] 218.5415

mean(VFdf4$speed)/1000 + qt(0.025, dim(VFdf4)[1] - 1)*sd(VFdf4$speed)/1000/sqrt(dim(VFdf4)[1] - 1)

#> [1] 133.0786

```

Using the approach above, the average movement speed is estimated as $176 \pm 43 \text{ km(Yr)}^{-1}$, an estimate that is less than the previous speed estimates for northeastern spread of 220 km(Yr)^{-1} (Bjørnstad et al. 2002) and 254 km(Yr)^{-1} (D. M. Johnson, Bjørnstad, and Liebhold 2004). The difference between estimates in the literature and estimates produced here are likely due to the direction of movement. The vectors in the larch budmoth vector field point predominantly north and east (Fig. 5). In other words they are orthogonal. In contrast the movement speeds estimated by Bjørnstad et al. (2002) and D. M. Johnson, Bjørnstad, and Liebhold (2004) are projected along lines that run to the northeast. A simple application of geometry reveals that an average speed of 176 km(Yr)^{-1} in the north and east directions corresponds to an estimated speed of 249 km(Yr)^{-1} in the northeastern direction (Pythagorean theorem $\sqrt{176^2 + 176^2}$). This estimate is consistent with prior speed estimates for larch budmoth population movement (Bjørnstad et al. 2002; D. M. Johnson, Bjørnstad, and Liebhold 2004).

4 Summary

The **ICvectorfields** R package implements standard Digital Image Correlation algorithms in addition to a novel extension that permits estimation of orthogonal velocities of persistent movement in series of three or more images. Here I demonstrate the usefulness of DIC and the extension implemented in **ICvectorfields** in a new arena: Whereas DIC is often applied in engineering and materials science to quantify the effects of force application on materials (Sutton, Orteu, and Schreier 2009), it has not been used in landscape ecology. In this field, the approach has potential to provide new insights into how populations move across landscapes and to demonstrate the untenable nature of assumptions of homogeneity inherent in most analyses based on the traveling wave paradigm. Even when models of sufficient complexity to capture environmental heterogeneity can be used, I expect that the methods in **ICvectorfields** will be useful because they facilitate comparison between modeled and empirical population movement data as demonstrated in the partial differential equation example in this study. Approaches such as this one that estimate movement based on cross-correlation, however, have a weakness: Under certain circumstances, they are prone to finding cross-correlations that are unrelated to movement as was demonstrated in this paper. For this reason, users must exercise vigilance in interpreting the results of vector field analyses like those demonstrated herein. If possible, results should be checked against a standard or against prior published results regarding movement propensity. Nevertheless, the methods described here hold promise for exploratory analyses, hypothesis generation, and synoptic pattern analyses of population movements.

5 Acknowledgements

I am grateful for constructive comments from two anonymous reviewers as well as for the patience of editors at The R Journal, which enabled this work navigate the peer review process.

References

- Alonso, David, Frederic Bartumeus, and Jordi Catalan. 2002. "Mutual Interference Between Predators Can Give Rise to Turing Spatial Patterns." *Ecology* 83 (1): 28–34.
- Anuta, Paul E. 1970. "Spatial Registration of Multispectral and Multitemporal Digital Imagery Using Fast Fourier Transform Techniques." *IEEE Transactions on Geoscience Electronics* 8 (4): 353–68.

- Avgar, Tal, Jonathan R Potts, Mark A Lewis, and Mark S Boyce. 2016. "Integrated Step Selection Analysis: Bridging the Gap Between Resource Selection and Animal Movement." *Methods in Ecology and Evolution* 7 (5): 619–30.
- Bjornstad, Ottar N. 2020. *Ncf: Spatial Covariance Functions*. <https://CRAN.R-project.org/package=nfc>.
- Bjørnstad, Ottar N, and Wilhelm Falck. 2001. "Nonparametric Spatial Covariance Functions: Estimation and Testing." *Environmental and Ecological Statistics* 8 (1): 53–70.
- Bjørnstad, Ottar N, Mikko Peltonen, Andrew M Liebhold, and Werner Baltensweiler. 2002. "Waves of Larch Budmoth Outbreaks in the European Alps." *Science* 298 (5595): 1020–23.
- Buderman, Frances E, Mevin B Hooten, Jacob S Ivan, and Tanya M Shenk. 2016. "A Functional Model for Characterizing Long-Distance Movement Behaviour." *Methods in Ecology and Evolution* 7 (3): 264–73.
- Campitelli, Elio. 2021a. *Ggnewscale: Multiple Fill and Colour Scales in 'Ggplot2'*. <https://CRAN.R-project.org/package=ggnewscale>.
- . 2021b. *metR: Tools for Easier Analysis of Meteorological Fields*. <https://doi.org/10.5281/zenodo.2593516>.
- Goodsmann, Devin. 2021. *ICvectorfields: Vector Fields from Spatial Time Series of Population Abundance*. <https://CRAN.R-project.org/package=ICvectorfields>.
- Hassell, Michael P, Hugh N Comins, and Robert M May. 1994. "Species Coexistence and Self-Organizing Spatial Dynamics." *Nature* 370 (6487): 290–92.
- Hassell, Michael P, Hugh N Comins, and Robert M May. 1991. "Spatial Structure and Chaos in Insect Population Dynamics." *Nature* 353 (6341): 255–58.
- Hijmans, Robert J. 2021. *Terra: Spatial Data Analysis*. <https://CRAN.R-project.org/package=terra>.
- Hoffman, Aaron L, Julian D Olden, Jeremy B Monroe, N LeRoy Poff, Todd Wellnitz, and John A Wiens. 2006. "Current Velocity and Habitat Patchiness Shape Stream Herbivore Movement." *Oikos* 115 (2): 358–68.
- Johnson, Derek M, Ottar N Bjørnstad, and Andrew M Liebhold. 2004. "Landscape Geometry and Travelling Waves in the Larch Budmoth." *Ecology Letters* 7 (10): 967–74.
- Johnson, Neil. 2009. *Simply Complexity: A Clear Guide to Complexity Theory*. Simon; Schuster.
- Joo, Rocio, Matthew E Boone, Thomas A Clay, Samantha C Patrick, Susana Clusella-Trullas, and Mathieu Basille. 2020. "Navigating Through the r Packages for Movement." *Journal of Animal Ecology* 89 (1): 248–67.
- Kolmogorov, Andrei N. 1937. "Étude de l'équation de La Diffusion Avec Croissance de La Quantité de Matière Et Son Application à Un Problème Biologique." *Bull. Univ. Moskow, Ser. Internat., Sec. A* 1: 1–25.
- Kot, Mark, Mark A Lewis, and Pauline van den Driessche. 1996. "Dispersal Data and the Spread of Invading Organisms." *Ecology* 77 (7): 2027–42.
- Krainski, Elias, Virgilio Gómez-Rubio, Haakon Bakka, Amanda Lenzi, Daniela Castro-Camilo, Daniel Simpson, Finn Lindgren, and Håvard Rue. 2018. *Advanced Spatial Modeling with Stochastic Partial Differential Equations Using r and INLA*. Chapman; Hall/CRC.
- Parrish, Julia K, and Leah Edelstein-Keshet. 1999. "Complexity, Pattern, and Evolutionary Trade-Offs in Animal Aggregation." *Science* 284 (5411): 99–101.
- R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.
- Rahim, Karim. 2021. *Eftwtools: Wrapper for 'Eftw3' Includes: One-Dimensional, Two-Dimensional, Three-Dimensional, and Multivariate Transforms*. <https://CRAN.R-project.org/package=eftwtools>.
- Ruan, SHIGUI. 1998. "Turing Instability and Travelling Waves in Diffusive Plankton Models with Delayed Nutrient Recycling." *IMA Journal of Applied Mathematics* 61 (1): 15–32.
- Signer, Johannes, John Fieberg, and Tal Avgar. 2019. "Animal Movement Tools (Amt): R Package for Managing Tracking Data and Conducting Habitat Selection Analyses." *Ecology and Evolution* 9: 880–90. <https://doi.org/10.1002/ece3.4823>.
- Skellam, John Gordon. 1951. "Random Dispersal in Theoretical Populations." *Biometrika* 38 (1/2): 196–218.
- Soetaert, Karline, and Filip Meysman. 2012. "Reactive Transport in Aquatic Ecosystems: Rapid Model Prototyping in the Open Source Software r." *Environmental Modelling & Software* 32: 49–60.
- Soetaert, Karline, Thomas Petzoldt, and R. Woodrow Setzer. 2010. "Solving Differential Equations in R: Package deSolve." *Journal of Statistical Software* 33 (9): 1–25. <https://doi.org/10.18637/jss.v033.i09>.
- Sutton, Michael A, Jean Jose Orteu, and Hubert Schreier. 2009. *Image Correlation for Shape, Motion and Deformation Measurements: Basic Concepts, Theory and Applications*. Springer Science & Business Media.
- Urban, Mark C, Ben L Phillips, David K Skelly, and Richard Shine. 2008. "A Toad More Traveled: The Heterogeneous Invasion Dynamics of Cane Toads in Australia." *The American Naturalist* 171 (3): E134–48.

- Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.
- Zammit-Mangion, Andrew. 2019. *IDE: Integro-Difference Equation Spatio-Temporal Models*. <https://CRAN.R-project.org/package=IDE>.

Bibliography

- M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2): 216–231, 2005. Special issue on “Program Generation, Optimization, and Platform Adaptation”. [p122]
- D. Goodsman. *ICvectorfields: Vector Fields from Spatial Time Series of Population Abundance*, 2021. URL <https://CRAN.R-project.org/package=ICvectorfields>. R package version 0.0.2. [p121]

Devin W. Goodsman
Natural Resources Canada
5320 122 St NW,
Edmonton, Alberta, T6H 3S5
Canada
NA
ORCID: 0000-0003-1935-5779
devin.goodsman@nrca-nrcan.gc.ca

Refreg: An R Package for Estimating Conditional Reference Regions

by Lado-Baleato, Óscar, Roca-Pardiñas, Javier, Cadarso-Suárez, Carmen and Gude, Francisco

Abstract Multivariate reference regions (MVR) represent the extension of the reference interval concept to the multivariate setting. A reference interval is defined by two threshold points between which a high percentage of healthy subjects' results, usually 95%, are contained. Analogously, an MVR characterizes the values of several diagnostic tests most frequently found among non-diseased subjects by defining a convex hull containing 95% of the results. MVRs have great applicability when working with diseases that are diagnosed via more than one continuous test, e.g., diabetes or hypothyroidism. The present work introduces **refreg**, an R package for estimating conditional MVRs. The reference region is non-parametrically estimated using a multivariate kernel density estimator, and its shape allowed to change under the influence of covariates. The effects of covariates on the multivariate variable means, and on their variance-covariance matrix, are estimated by flexible additive predictors. Continuous covariate non-linear effects can be estimated by penalized spline smoothers. The package allows the user to propose, for instance, an age-specific diagnostic rule based on the joint distribution of two non-Gaussian, continuous test results. The usefulness of the **refreg** package in clinical practice is illustrated with a real case in diabetes research, with an age-specific reference region proposed for the joint interpretation of two glycemia markers (fasting plasma glucose and glycated hemoglobin). To show that the **refreg** package can also be used in other, and indeed very different fields, an example is provided for the joint prediction of two atmospheric pollutants (SO_2 , and NO_x). Additionally, the text discusses how, conceptually, this method could be extended to more than two dimensions.

1 Introduction

In clinical practice, many medical decisions are based on continuous diagnostic tests (Hallworth, 2011) – i.e., tests that provide results along a continuous, quantitative scale. The interpretation of such continuous tests requires the comparison of the obtained value with a pre-defined reference interval, so that a result could be classified as positive or negative (ie, disease present or absent) based on this comparator value. A reference interval is an interval containing most healthy subjects' results. For a single test they are usually estimated from the 2.5 and 97.5 empirical percentiles of the distribution for the healthy population; thus, 95% of healthy patients are located within the interval limits (Wright and Royston, 1999). Those patients falling outside the reference interval, are likely to have an undiagnosed disease. If the test results are influenced by some patient characteristics independent of the disease (e.g., age and gender), reference intervals for specific patient groups must be obtained. These covariate-dependent reference intervals, usually termed reference curves, are estimated using quantile regression (Koenker and Bassett Jr, 1978) or location-scale models (Cole and Green, 1992; Stasinopoulos et al., 2017). Several R packages for estimating reference intervals and reference curves already exist, including the R package **referenceIntervals**, which comprises a collection of tools, the R package **gamlss** (Stasinopoulos et al., 2007), which provides a general tool for deriving reference curves in clinical practice (WHO, 2006), and software *RefCurv* (Winkler et al., 2019), recently proposed to facilitate clinicians' use of **gamlss**. However, all these packages were produced to provide reference intervals for single tests; they cannot address diseases for which diagnosis and control are based on multiple tests.

When the results of several tests are available for the same patient, obtaining separate reference intervals for each one provides an incomplete picture of disease status, particularly when these results are strongly correlated (Boyd, 2004). Although each reference interval would leave only 5% of healthy patients out, their combined use can result in a higher percentage of false positives. Moreover, a patient falling within each univariate reference interval might, in fact, show an abnormal multivariate result. Thus, a multivariate reference region (MVRs) would provide a better means of interpreting the results of multiple tests (Winkel and Lyngbye, 1972). MVRs are a straightforward extension of the univariate reference interval to the multidimensional setting, i.e., a region that contains 95% of the healthy patients' results. However, despite being proposed more than 40 years ago, MVRs are rarely used in clinical practice. This might be explained by the multivariate Gaussian assumption of MVRs, which is quite restrictive when interpreting diagnostic test results. Further, the multivariate distribution of test results is usually affected by patient characteristics, independent of their health status. For example, Espasandín-Domínguez et al. (2019) showed that the correlation between two diagnostic tests for diabetes was influenced by patient age and red blood cell turnover, independent of glycemia status. A conditional MVR is therefore desirable, but the statistical literature is not rich in such proposals (see, e.g., (Wei, 2008)).

Very few statistical software routines have been proposed for estimating the region containing a specific percentage of multivariate data points. The function `mvto1.region` in the R package **tolerance** (Young, 2010) obtains a region containing a high percentage of a bivariate Gaussian distribution in the context of quality control studies, and non-parametric probabilistic regions can be obtained using the R packages **r2d2** (Magnusson and Burgos, 2014), **hdcrcde** (Hyndman, 2018) and **distfree.cr** (Hu and cai Yang, 2018). However, these all suffer the major limitation of not being able to estimate the effects of covariates on the region's shape. The R package **modQR** can estimate conditional multivariate quantiles, but the quantile level τ is not linked to the probability content of the sample (Šiman and Boček, 2019). Thus, it is not clear how to derive a reference region from these bivariate quantiles.

The present paper introduces **refreg**, an implementation in R of a new statistical methodology for estimating bivariate reference regions able to classify subjects as having normal or abnormal values based on the results of two continuous diagnostic tests. The main advantages of the presented method are: i) the absence of parametric restrictions for describing bivariate distributions for continuous tests, and ii) the possibility of estimating the effects of covariates on the shape of the reference region via flexible additive predictors. To illustrate this statistical methodology, and how to use the package, an age-specific reference region for two diabetes diagnostic tests was estimated. The estimated reference region offers new insights into the diagnosis and prognosis of diabetes, enabling physicians to identify different patients' profiles. The proposed method can, however, be applied to any disease in which two continuous diagnostic tests are available, and can even be used in non-medical fields. Indeed, an application is discussed in which the conditional region is used in the joint forecasting of the concentrations of two air pollutants. Moreover, the current implementation can be easily extended to three dimensions.

The statistical model that enables conditional reference regions to be determined is presented in the next section. The main functions contained in the **refreg** package are then described, with a brief introduction to the main functions. The use of the package is shown in analyses of real medical and environmental data problems. The paper closes with some comments, and some notes on future research directions.

2 Statistical methodology

In this section the main features of our statistical method is presented. In a nutshell, our proposal is based on a bivariate location scale model, where the response means, and their variance-covariance matrix, are related to covariates using flexible additive predictors. The probabilistic region covering a specific percentage of the data is firstly estimated using the model standardized residuals. Then, it is generalized for each covariate value based on the aforementioned bivariate location-scale model fit. This statistical model was already presented and evaluated in (Roca-Pardiñas et al., 2020).

Conditional reference region

Given a bivariate continuous random variable of interest $\mathbf{Y} = (Y_1, Y_2)$, and a vector of covariates $\mathbf{X} = (X_1, \dots, X_p)$ we consider the following structure:

$$\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} \mu_1(\mathbf{X}) \\ \mu_2(\mathbf{X}) \end{pmatrix} + \boldsymbol{\Sigma}^{1/2}(\mathbf{X}) \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \end{pmatrix} \quad (1)$$

where $\mu_1(\mathbf{X})$ and $\mu_2(\mathbf{X})$ represents the conditional means of both responses and $\boldsymbol{\Sigma}^{1/2}(\mathbf{X})$ the Cholesky decomposition of the variance-covariance matrix

$$\boldsymbol{\Sigma}(\mathbf{X}) = \begin{pmatrix} \sigma_1^2(\mathbf{X}) & \sigma_{12}(\mathbf{X}) \\ \sigma_{12}(\mathbf{X}) & \sigma_2^2(\mathbf{X}) \end{pmatrix} \quad (2)$$

so that $\text{Var}(\mathbf{Y}|\mathbf{X}) = \boldsymbol{\Sigma}(\mathbf{X}) = \boldsymbol{\Sigma}^{1/2}(\mathbf{X}) \left(\boldsymbol{\Sigma}^{1/2}(\mathbf{X}) \right)^T$. In order to guarantee the model identification (1), the bivariate residuals $(\varepsilon_1, \varepsilon_2)$ are assumed to be independent of the covariates, with zero mean, unit variance, and zero correlation.

We consider additive structures for the mean functions $\mu_j(\mathbf{X})$, variance functions $\sigma_j^2(\mathbf{X})$ ($j = 1, 2$) and the correlation function $\rho(\mathbf{X})$ – note that $\sigma_{12}(\mathbf{X}) = \delta_1(\mathbf{X})\delta_2(\mathbf{X})\rho(\mathbf{X})$. These structures are given, respectively, by

$$\mu_r(\mathbf{X}) = \alpha_r + \sum_{j=1}^p f_{jr}(X_j), \quad \sigma_r^2(\mathbf{X}) = H_\sigma \left(\beta_r + \sum_{j=1}^p g_{jr}(X_j) \right) \quad \text{for } r = 1, 2$$

and

$$\rho(\mathbf{X}) = H_\rho \left(\gamma + \sum_{j=1}^p m_j(X_j) \right)$$

where α, β and γ are parametric coefficients and f_{jr}, g_{jr} and m_j for $j = 1, \dots, p$ and $r = 1, 2$ are smooth and unknown functions. $H_\sigma(\cdot) = \exp(\cdot)$ and $H_\rho(\cdot) = \tanh(\cdot)$ are link functions used in the variance and correlation structures, respectively, to ensure that the restrictions on the parameter spaces ($\sigma_r^2(\mathbf{X}) \geq 0$ and $0 \leq \rho(\mathbf{X}) \leq 1$) are maintained.

Based on the model presented in equation (1), for a given \mathbf{X} the conditional τ^{th} - reference region for (Y_1, Y_2) is given by:

$$R_\tau(\mathbf{X}) = \begin{pmatrix} \mu_1(\mathbf{X}) \\ \mu_2(\mathbf{X}) \end{pmatrix} + \Sigma^{1/2}(\mathbf{X})\varepsilon_\tau \tag{3}$$

where ε_τ is the unconditionally probabilistic region for the errors $(\varepsilon_1, \varepsilon_2)$ as

$$\varepsilon_\tau(k) = \{(\varepsilon_1, \varepsilon_2) \in R^2 | f(\varepsilon_1, \varepsilon_2) \leq k\} \tag{4}$$

f being the density function of the bivariate residuals $(\varepsilon_1, \varepsilon_2)$ and k is the τ -quantile of $f(\varepsilon_1, \varepsilon_2)$.

Estimation algorithm

In this section, we present the estimation procedure of the conditioned bivariate uncertainty region given in equation (3). Our approach is based on the estimation of the covariate effects on the response means using an additive model, and then on the variance-covariance matrix using the squared residuals of the former models. Finally, the bivariate region $R_\tau(\mathbf{X})$ is obtained with a bivariate kernel estimation of the standardized bivariate residuals.

The steps of the proposed estimation algorithm are the following:

Step 1: For $r = 1, 2$ fit an additive model to the sample $\{\mathbf{X}_i, \mathbf{Y}_{ir}\}_{i=1}^n$ and obtain the estimates

$$\hat{\mu}_r(\mathbf{X}_i) = \hat{\alpha} + \sum_{j=1}^p \hat{f}_{jr}(X_{ij}) \tag{5}$$

Then, estimate $\hat{\sigma}_r^2(\mathbf{X})$ from the sample $\{\mathbf{X}_i, (\mathbf{Y}_{ir} - \hat{\mu}_r(\mathbf{X}_i))^2\}_{i=1}^n$ as

$$\hat{\sigma}_r^2(\mathbf{X}_i) = \hat{\beta}_r + \sum_{j=1}^p \hat{g}_{jr}(X_{ij}) \tag{6}$$

Step 2: Compute the correlation $\rho(\mathbf{X})$, using the sample $\{\mathbf{X}_i, \hat{\delta}_i\}_{i=1}^n$, as

$$\hat{\rho}(\mathbf{X}_i) = \tanh \left(\hat{\gamma} + \sum_{j=1}^p \hat{m}_{jr}(X_{ij}) \right)$$

where

$$\hat{\delta}_i = \frac{(Y_{i1} - \hat{\mu}_1(\mathbf{X}_i))(Y_{i2} - \hat{\mu}_2(\mathbf{X}_i))}{\hat{\sigma}_1(\mathbf{X}_i)\hat{\sigma}_2(\mathbf{X}_i)}$$

Step 3: Compute the standardized residuals

$$\begin{pmatrix} \hat{\varepsilon}_{i1} \\ \hat{\varepsilon}_{i2} \end{pmatrix} = \hat{\Sigma}^{-1/2}(\mathbf{X}_i) \begin{pmatrix} Y_{i1} - \hat{\mu}_1(\mathbf{X}_i) \\ Y_{i2} - \hat{\mu}_2(\mathbf{X}_i) \end{pmatrix} \quad i = 1, \dots, n \tag{7}$$

and obtain the kernel estimation of the bivariate density $\hat{f}(\varepsilon_1, \varepsilon_2)$ given by

$$\hat{f}((\varepsilon_1, \varepsilon_2), \mathbf{H}) = \frac{1}{n} \sum_{i=1}^n K_{\mathbf{H}} \begin{pmatrix} \varepsilon_1 - \hat{\varepsilon}_{i1} \\ \varepsilon_2 - \hat{\varepsilon}_{i2} \end{pmatrix} \tag{8}$$

where $K(\cdot)$ is the kernel which is a symmetric probability density function and \mathbf{H} is a 2×2 positive definite matrix. Then, obtain the τ^{th} unconditional bivariate uncertainty region on the residual scale as

$$\hat{\varepsilon}_\tau = \{(\varepsilon_1, \varepsilon_2) \in R^2 | \hat{f}(\varepsilon_1, \varepsilon_2) \leq \hat{k}\} \tag{9}$$

\hat{k} being the empirical τ quantile of the values $\hat{f}(\varepsilon_{11}, \varepsilon_{12}), \dots, \hat{f}(\varepsilon_{n1}, \varepsilon_{n2})$.

Finally, for a given \mathbf{X} , the conditional bivariate uncertainty region $R_\tau(\mathbf{X})$ is estimated by

$$\hat{R}_\tau(\mathbf{X}) = \begin{pmatrix} \hat{\mu}_1(\mathbf{X}) \\ \hat{\mu}_2(\mathbf{X}) \end{pmatrix} + \hat{\Sigma}^{1/2}(\mathbf{X})\hat{\varepsilon}_\tau \tag{10}$$

Flexible additive models estimation and inference

The continuous covariates smooth effects (f_{jr} , g_{jr} and m_j for $j = 1, \dots, p$) may be estimated using several non-parametric regression techniques. In previous works we applied polynomial kernel smoothers (Roca-Pardiñas et al., 2020). However, for sake of usage simplicity and computational cost, in the final package implementation we used a penalized spline basis representation following (Wood, 2017). Thus, given an unknown smooth effect (e.g. $f(x)$) is estimated as:

$$f(x) = \sum_{k=1}^K \beta_k b_k(x)$$

Confidence intervals for the estimated effects may be obtained using a bootstrap procedure. Given a specific vector of covariates \mathbf{X}_0 , for the components (means, variances and correlation). The steps for construction of the bootstrap confidence intervals are:

Step 1. From the sample data $\{(Y_{i1}, Y_{i2}), \mathbf{X}_i\}_{i=1}^n$ obtain the estimates $\hat{\mu}_r(\mathbf{X}_0)$, $\hat{\sigma}_r(\mathbf{X}_0)$ ($r = 1, 2$) and $\hat{\rho}(\mathbf{X}_0)$.

Step 2. For $b = 1, \dots, B$ generate bootstrap samples $\{(Y_{i1}^\bullet, Y_{i2}^\bullet), \mathbf{X}_i\}_{i=1}^n$ with

$$\begin{pmatrix} Y_{i1}^\bullet \\ Y_{i2}^\bullet \end{pmatrix} = \begin{pmatrix} \hat{\mu}_1(\mathbf{X}_i) \\ \hat{\mu}_2(\mathbf{X}_i) \end{pmatrix} + \hat{\Sigma}^{1/2}(\mathbf{X}_i) \begin{pmatrix} \hat{\varepsilon}_{i1}^\bullet \\ \hat{\varepsilon}_{i2}^\bullet \end{pmatrix} \tag{11}$$

where $\{(\hat{\varepsilon}_{i1}^\bullet, \hat{\varepsilon}_{i2}^\bullet)\}_{i=1}^n$ is a sample of size n from the residuals $\{(\hat{\varepsilon}_{i1}, \hat{\varepsilon}_{i2})\}_{i=1}^n$ with replacement, and compute $\hat{\mu}_r^{\bullet b}(\mathbf{X}_0)$, $\hat{\sigma}_r^{\bullet b}(\mathbf{X}_0)$ and $\hat{\rho}^{\bullet b}(\mathbf{X}_0)$ as in Step 1.

The limits for the $100(1 - \alpha)\%$ confidence intervals of the true components $\mu_r(\mathbf{X}_0)$, $\sigma_r(\mathbf{X}_0)$ and $\rho(\mathbf{X}_0)$ are given respectively by $(\hat{\mu}_r^{\alpha/2}(\mathbf{X}_0), \hat{\mu}_r^{1-\alpha/2}(\mathbf{X}_0))$, $(\hat{\sigma}_r^{\alpha/2}(\mathbf{X}_0), \hat{\sigma}_r^{1-\alpha/2}(\mathbf{X}_0))$ and $(\hat{\rho}^{\alpha/2}(\mathbf{X}_0), \hat{\rho}^{1-\alpha/2}(\mathbf{X}_0))$, where $\hat{\mu}_r^p(\mathbf{X}_0)$ represents the p -percentile of $\hat{\mu}_r^{\bullet 1}(\mathbf{X}_0), \dots, \hat{\mu}_r^{\bullet B}(\mathbf{X}_0)$, $\hat{\sigma}_r^p(\mathbf{X}_0)$ represents the p -percentile of $\hat{\sigma}_r^{\bullet 1}(\mathbf{X}_0), \dots, \hat{\sigma}_r^{\bullet B}(\mathbf{X}_0)$, and $\hat{\rho}^p(\mathbf{X}_0)$ is the p -percentile of $\hat{\rho}^{\bullet 1}(\mathbf{X}_0), \dots, \hat{\rho}^{\bullet B}(\mathbf{X}_0)$.

Bivariate residuals density estimation

The estimation of the unconditionally probabilistic region for the bivariate errors $(\varepsilon_1, \varepsilon_2)$ is based on a kernel density estimator. This estimator is given by:

$$\hat{f}((\varepsilon_1, \varepsilon_2), \mathbf{H}) = \frac{1}{n} \sum_{i=1}^n K_{\mathbf{H}} \begin{pmatrix} \varepsilon_1 - \hat{\varepsilon}_{i1} \\ \varepsilon_2 - \hat{\varepsilon}_{i2} \end{pmatrix} \tag{12}$$

where \mathbf{H} is a matrix defining the kernel bandwidth

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} \\ h_{12} & h_{22} \end{pmatrix}$$

The selection of \mathbf{H} is crucial to obtain a good estimation of ε_τ . A natural option is to use a plug-in or cross-validation bandwidth estimator, as in a density estimation problem

$$\arg \min_{\mathbf{H} \in \mathcal{H}} E \left(\iint (\hat{f}_{\mathbf{H}}(\varepsilon_1, \varepsilon_2) - f_{\mathbf{H}}(\varepsilon_1, \varepsilon_2)^{(-i)})^2 d\varepsilon_1 d\varepsilon_2 \right) \tag{13}$$

where $\hat{f}_{\mathbf{H}}(\varepsilon_1, \varepsilon_2)^{-i}$ is the estimated bivariate density function without the i -th observation. However, as we seek for a probabilistic region (i.e. a region which contains a given percentage of the multivariate data), the following selection criteria based on the region coverage is proposed

$$\hat{\lambda} = \arg \min_{\lambda} \left| \left(n^{-1} \sum_{i=1}^n I\{(Y_{i1}, Y_{i2}) \in R^{(-i)}(\mathbf{X}_i)\} \right) - \tau \right| \tag{14}$$

where τ is the desired coverage and $R^{(-i)}(\mathbf{X}_i)$ is the estimated bivariate region without the i -th observation. Using this criteria the estimated region show a smoother contour and a coverage of

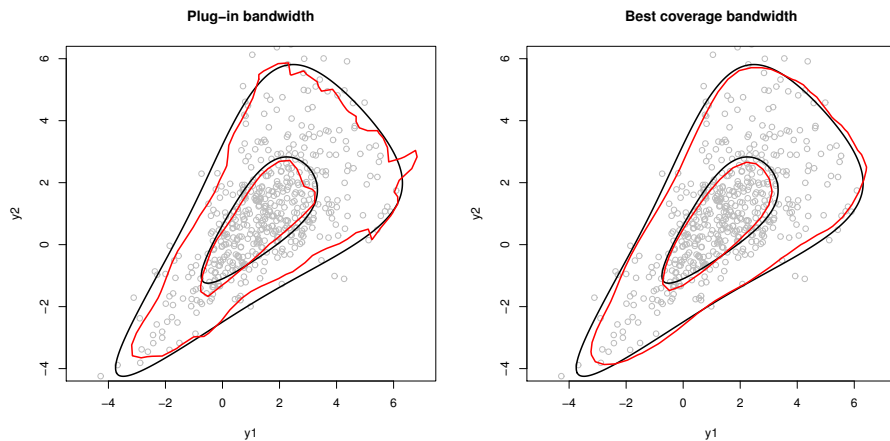


Figure 1: Estimated reference regions for plug-in and best coverage bandwidths for non-gaussian data. In black we represent true regions and in red the estimated ones for $\tau = 0.50, 0.95$. Best coverage bandwidth offers a smoother region than plug-in estimator.

the data points closer to the desired one τ (see Figure 1). Given the high computational cost of the regular proposed method in (13), a k-fold cross-validation scheme could be used instead. Moreover we simplify the minimization problem by considering $h_{11} = h_{22}$ and $h_{12} = 0$.

3 Overview of the package

The `refreg` package contains a set of functions for estimating a conditional reference, or uncertainty, region. Its working framework was designed so that people without a strong statistical background can use it. Indeed, only two functions need to be taken into account by the user: 1) the effects of the predictor variables on responses need to be estimated using the `bivRegr` function, a step that requires the user choose which variables may influence the region; 2) `bivRegion` needs to be applied to a `bivRegr` object so that the reference region can be estimated.

The `bivRegr()` function has the following structure:

```
bivRegr(f = formulas, data = data)
```

The `f` argument contains a list of five R formulae corresponding to the additive predictors for the means, variances and correlation models shown in equation (1). Since `bivRegr()` uses `mgcv::gam()` internally, the user can estimate covariate linear and non-linear effects using `s()` operator. For instance:

```
mu1 <- y1 ~ s(x1)
mu2 <- y2 ~ s(x1)
var1 <- ~ x2
var2 <- ~ x2
rho <- ~ s(x3)
```

```
formula = list(mu1, mu2, var1, var2, rho)
```

assumes a smooth effect of `x1` on the response means, a parametric effect of `x2` on their variances, and a smooth effect of `x3` on the response correlation.

The `bivRegion()` function is designed for non-parametrically estimating a bivariate reference region:

```
bivRegion(object, tau = 0.95, bandwidth = "plug-in")
```

The `object` may be a set of bivariate data points, or a `bivRegr` object, while `tau` defines the desired coverage(s) for the reference region, which might be a single value or a vector. Finally, “bandwidth” specifies the kernel bandwidth selection method. The user can chose between the plug-in, cross-validation, or the best coverage method (see equation (13)).

Additionally, we defined S3 methods for these two main functions. Specifically, associated with `bivRegr` we have

- `predict.bivRegr` and `plot.bivRegr`: to predict and depict additive models results for responses' means, variances, and their correlation.
- `summary_boot.bivRegr`: a function implementing the bootstrap inference for flexible additive models (see (11)). This function results can be depicted by applying `plot.summary_boot`.

On the other hand, we defined the following S3 methods associated to `bivRegion`:

- `summary.bivRegion`: this function evaluates the region performance on the healthy patients' sample.
- `predict.bivRegion` and `plot.bivRegion`: these functions offer a prediction or a plot of conditional regions for a new dataset. If the argument `cond=FALSE` it evaluates the response values in the standardized scale.

In addition, we define the functions `trivRegr`, `trivRegion` and `plot.trivRegion` as an extension of the aforementioned method for a trivariate response variable. Finally, our package also contains some inner functions as `ace` (for estimating variance, and correlation models), `Hcv` (it implements equation (14) method), and `refcurve` (it implements an univariate location-scale model).

4 Refreg in practice

This section outlines the implemented functions of the proposed package in detail, and illustrates their use with real datasets. The first illustration is related to diabetes research, in which a reference region for the joint interpretation of two glycemia tests is calculated. In the second illustration, `refreg` methodology is used to predict the concentrations of two air pollutants during a pollution episode. Finally, the extension of the method to higher dimensions is shown using real data.

Case 1: Glycemic tests for diabetes diagnosis

Diabetes is a chronic disease, the diagnosis of which is based on two glycemia tests: the fasting plasma glucose (FPG) and glycated hemoglobin (HbA1c) ([American Diabetes Association 2019](#)) tests. The multivariate interpretation of FPG and HbA1c results is desirable for two reasons: i) the results of both tests are correlated in healthy patients ([Aleyassine et al., 1980](#)), ii) a miss-match between them may be indicative of a poorer prognosis ([Kim et al., 2018](#)). Finally, it is well known that both test results are influenced by patient age ([Davidson, 1979](#); [Pani et al., 2008](#)).

The age-dependent reference region for the FPG and HbA1c tests was estimated using a sample of healthy subjects derived from the A-Estrada Glycation and Inflammation Study (AEGIS) (see ([Gude et al., 2017](#))). A subset of this dataset is available in the package under the name "AEGIS".

This dataset comprised 1516 subjects and 7 variables:

- `id`: an anonymous identifier for each subject.
- `gender`: a factor variable that indicates the subject's gender with levels "male", and "female".
- `age`: the subject's age.
- `dm`: a factor variable indicating a previous diabetes mellitus diagnosis with levels "no", and "yes".
- `fpg`: fasting plasma glucose concentration in mg/dL.
- `hba1c`: the percentage of glycated hemoglobin.
- `fru`: fructosamine plasma concentration.

Applying the `summary()` routine to the `aegis` dataset indicated 55% of the subjects to be female, the mean age of all 1516 subjects to be 52 years (range 18-91), and that 187 subjects (12%) had been previously diagnosed with diabetes.

```
R> summary(aegis)
      id      gender      age      dm      fpg
Min.   : 1.0  female:838  Min.   :18.00  no :1329  Min.   : 63.00
1st Qu.:379.8  male :678    1st Qu.:39.00  yes: 187  1st Qu.: 82.00
Median :758.5                Median :52.00                Median : 89.00
Mean   :758.5                Mean   :52.58                Mean   : 94.51
3rd Qu.:1137.2              3rd Qu.:67.00              3rd Qu.:100.00
Max.   :1516.0              Max.   :91.00                Max.   :274.00
```



```

hba1c      fru
Min.   : 3.900  Min.   :119.0
1st Qu.: 5.200  1st Qu.:225.0
Median : 5.400  Median :254.0
Mean   : 5.608  Mean   :262.2
3rd Qu.: 5.700  3rd Qu.:284.0
Max.   :10.200  Max.   :700.0

```

To estimate the reference region, a subset of the patients not previously diagnosed with diabetes was define as `dm_no`. This subset sample was deemed the healthy patient sample.

```

R> dm_no = subset(aegis,aegis$dm == "no")
R> dm_yes = subset(aegis,aegis$dm == "yes")

```

To estimate the effect of age on the final region shape, the `bivRegr()` function was used. This function implements the estimation process of the bivariate location-scale:

```

R> mu1 = fpg ~ s(age)
R> mu2 = hba1c ~ s(age)
R> var1 = ~ s(age)
R> var2 = ~ s(age)
R> rho = ~ s(age)
R> formula = list(mu1,mu2,var1,var2,rho)

```

The first and second formulae define the additive models for the mean values of both glycemia tests. The third and fourth define the additive models for test result variability. The last formula represents the additive model that comprises the effect of age on the correlation between the results of both glycemia tests. In addition to the model formulae list, a dataset including both the test results and subject's age must be supplied to `bivRegr()` as:

```

R> fit = bivRegr(formula,data=dm_no)

```

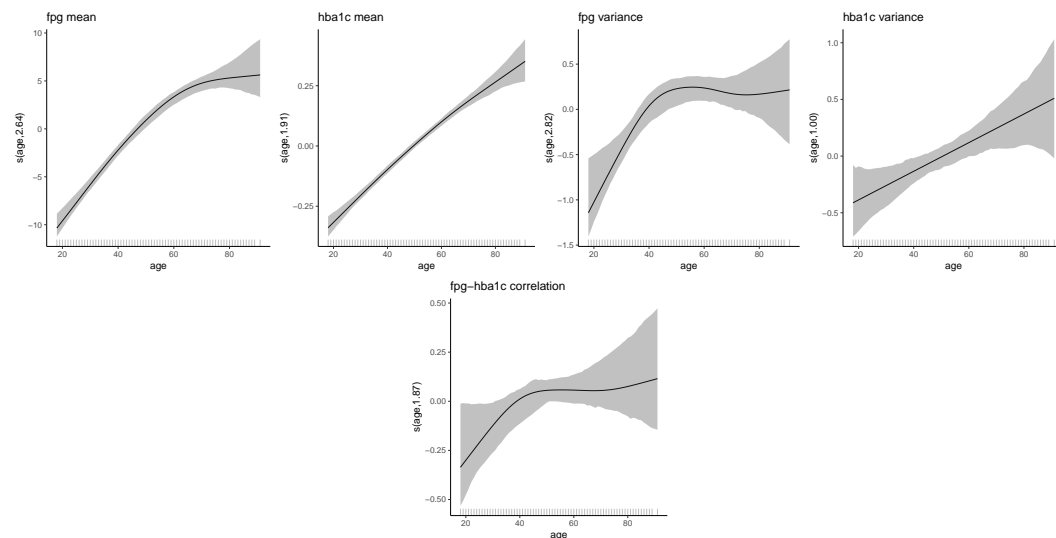


Figure 2: Estimated effects of age on the FPG and HbA1c mean, variance models, and on their correlation. Output from `summary_boot`, the shaded area is the 95% pointwise confidence interval obtained by bootstrap resampling. The parameters of our bivariate response change with age.

By applying the `S3` method `plot()` to a `bivRegr` object, the estimated effects of covariates can be shown for each submodel. The argument `eq=` controls the model component to be represented (1 = FPG mean, 2 = HbA1c mean, 3 = FPG variance, 4 = HbA1c variance, and 5 = [FPG – HbA1c] correlation). Moreover, the function `summary_boot()` may be applied to a `bivRegr` object to obtain the 95% pointwise confidence interval for the estimated effects via bootstrapping:

```

R> fit_boot = summary_boot(fit, B=250, parallel = TRUE )
R> plot(fit_boot,eq=1)
R> plot(fit_boot,eq=2)

```

```
R> plot(fit_boot,eq=3)
R> plot(fit_boot,eq=4)
R> plot(fit_boot,eq=5)
```

Since bootstrap resampling (introduced in equation (11)) is time consuming, the user can fix `parallel = TRUE` and run a parallelized computation. The parallel backend is registered using `doParallel` (Microsoft and Weston, 2020), and the parallel computation is performed by `foreach` (Microsoft and Weston, 2020).

Figure 2 shows that the mean values of both glycemia markers increase almost linearly with age. FPG variance increases from 20 to 40 years, while the HbA1c variance increases linearly with age. Finally, the correlation between the FPG and HbA1c concentration seems to be stronger for older patients.

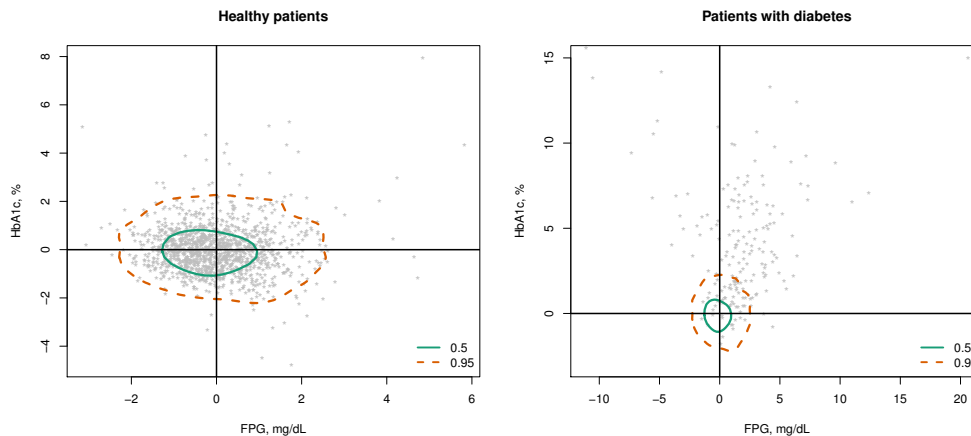


Figure 3: Estimated region in the bivariate residuals scale for healthy patients (left), and patients with diabetes (right). Green contour represents the region for $\tau = 0.50$, while red dashed contour for $\tau = 0.95$.

Applying the function `bivRegion()` to a `bivRegr` object provides a bivariate region containing $100\tau\%$ of the model standardized residuals. This region is based on a bivariate kernel density estimator. The kernel bandwidth selection method may be chosen with the `H_choice` argument. Here, the 90%, 95% and 97.5% regions are obtained with the best coverage bandwidth selector (see equation (13)):

```
R> region = bivRegion(fit,tau=c(0.90,0.95,0.975),H_choice = "Hcov")
```

This region facilitates a multivariate interpretation of the glycemia test results. A patient whose results are “normal”, for his/her age would see them fall inside this reference region, while a subject with “abnormal” results for his/her age would not. This interpretation is possible because the model residuals are centered around zero, show unit variance, no correlation, and they are independent of age. The user can check test results located outside the reference region using the `bivRegion S3` method plot:

```
R> par(mfrow = c(1, 2))
R> plot(region, xlab = "FPG, mg/dL", ylab = "HbA1c, %",cond=T, newdata =
  data.frame (age = c(20,30,40,50,60,70)),tau=0.95,reg.lwd=2, pch="*",col="grey")

R> plot(region,xlab = "FPG, mg/dL", ylab = "HbA1c, %",cond=T, newdata =
  data.frame(age = c(20,60)),tau=c(0.50,0.95), reg.lwd=2, pch="*", col="grey")
```

Figure 3 shows the unconditional reference region for $\tau = 0.90, 0.95$ and 0.975 for healthy patients, and those previously diagnosed with diabetes. Note that the `plot()` function argument ‘`newdata`’ allows the glycemia test values to be observed in the standardized residuals scale of the dataset for the patients with diabetes. As is clearly seen, most of healthy patients’ results are located inside the reference region, while those recorded for diabetic patients are located outside.

A major advantage of this representation is that it allows clinicians new insights into the subject’s glycemia status. Indeed, those patients located outside the reference region may be classified into four groups: (I) individuals with high values for both tests (first quadrant); (II) those with discordant results,

with high HbA1c concentrations and low/medium FPG (second quadrant); (III) individuals with low values for both tests (third quadrant); and (IV) individuals with low/medium HbA1c concentrations and high FPG values (fourth quadrant). This distinction might be useful for physicians. For instance, discordant results are probably due to an altered bloodstream protein glycation rate, a condition associated with a poorer prognosis. Patients located outside the standardized region may be also checked applying `summary()` to a `bivRegion` object:

```
R> summary(region,tau = 0.95)
```

This R output presents patients located outside the standardized bivariate region for $\tau = 0.95$. Note that, in the full sample, patients with different ages were located outside the reference region. The glycemia tests results located outside the reference region are interesting from a clinical point of view. For example, the following were seen: a possible case of undiagnosed diabetes in a 20 year old patient (FPG = 99, HbA1c = 6.3); a 47 year old patient showing a high HbA1c value for his corresponding FPG result (FPG = 86, HbA1c = 6); and a patient of 85 years in the opposite situation (FPG = 120, HbA1c = 5.7).

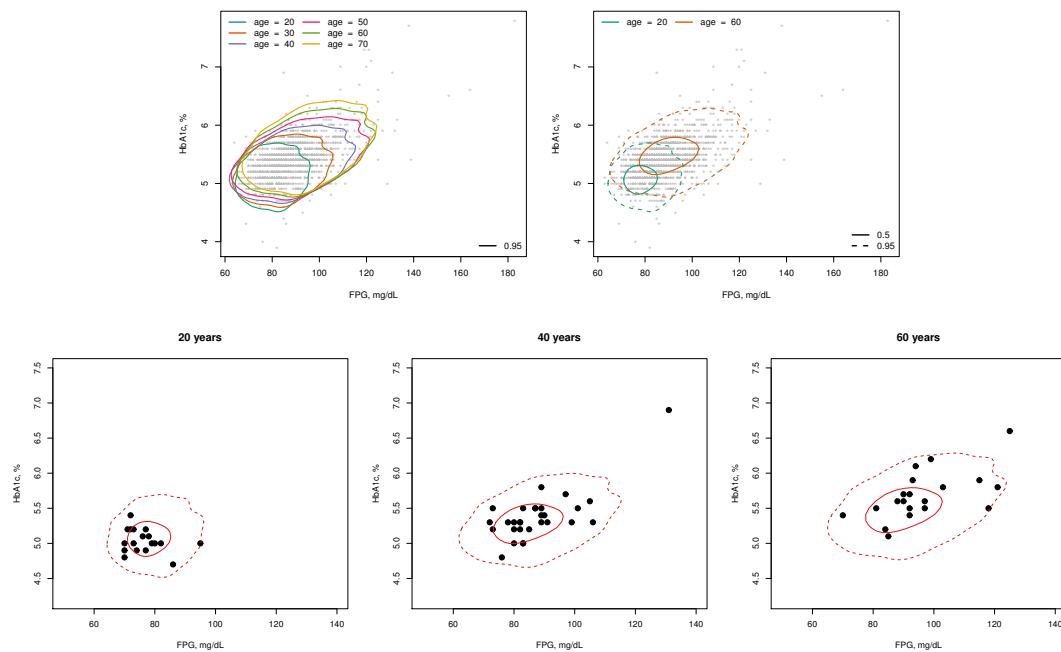


Figure 4: Predicted reference regions for different ages. Solid line contour represents the reference region for $\tau = 0.50$, and the dashed line contour for $\tau = 0.95$. Toprow plots are depicted by `plot.bivRegion` function setting the arguments `cond=TRUE` and `add=FALSE` for several ages, and bottomrow ones with `cond=TRUE` and `add=TRUE` in a pre-existing plot. The estimated regions change with age and it describes the shape of the observed data points.

The use of this region in combination with the results of the bivariate location-scale model allow the conditional reference regions to be obtained. The user can visualize these regions using the `bivRegion` S3 method `plot()`, setting `cond = TRUE` as follows:

```
R> plot(region, xlab = "FPG, mg/dL", ylab = "HbA1c, %", cond=T, newdata = data.frame(age = c(20,30,40,50,60,70)), tau=0.95, reg.lwd=2, pch="*", col="grey")
```

```
R> plot(region,xlab = "FPG, mg/dL", ylab = "HbA1c, %", cond=T, newdata = data.frame(age = c(20,60)), tau=c(0.50,0.95), reg.lwd=2, pch="*", col="grey")
```

In addition, the region may be represented in a pre-existing plot if the `plot` function argument `add` is equal to `TRUE` as in the following code:

```
R> plot(dm_no[dm_no$age==40,"fpg"],dm_no[dm_no$age==40,"hba1c"],main="40 years",
      xlim=c(50,140), ylim=c(4.2,7.5),xlab = "FPG, mg/dL", ylab = "HbA1c, %", pch=20,cex=2)
R> plot(region,cond=T,newdata = data.frame(age = 40),add=T,legend=F, tau=c(0.50,0.95),
      reg.lty=c(1,2))
```

```
R> plot(dm_no[dm_no$age==60,"fpg"],dm_no[dm_no$age==60,"hba1c"],main="60 years",
```

```
xlim=c(50,140), ylim=c(4.2,7.5), xlab = "FPG, mg/dL", ylab = "HbA1c, %", pch=20, cex=2)
R> plot(region, cond=T, newdata = data.frame(age = 60), add=T, legend=F, tau=c(0.50,0.95),
reg.lty=c(1,2))
```

In Figure 4, the bivariate reference region is shown for several ages. Note that the regions shift towards the upper right corner and expand as age increase. This agrees with the non-linear effect of age on the expected means and variability of both markers. The conditional region coverage and the performance of the methodology have already been assessed (Lado-Baleato et al., 2021).

Extension of case 1, conditional reference region for a trivariate response

This section provides an example of how the method proposed in equation (3) might be extended to more than two dimensions. This section is intended to provide a proof of concept rather than a formal statistical proposal. For a trivariate variable (Y_1, Y_2, Y_3) the following model can be assumed:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} \mu_1(\mathbf{X}) \\ \mu_2(\mathbf{X}) \\ \mu_3(\mathbf{X}) \end{pmatrix} + \Sigma^{1/2}(\mathbf{X}) \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{pmatrix} \tag{15}$$

where $\{\mu_r(\mathbf{X})\}_{r=1}^3$ represents the conditional means of each response, and $\Sigma^{1/2}(\mathbf{X})$ the Cholesky decomposition of the variance-covariance matrix

$$\Sigma(\mathbf{X}) = \begin{pmatrix} \sigma_1^2(\mathbf{X}) & \sigma_{21}(\mathbf{X}) & \sigma_{31}(\mathbf{X}) \\ \sigma_{12}(\mathbf{X}) & \sigma_2^2(\mathbf{X}) & \sigma_{23}(\mathbf{X}) \\ \sigma_{13}(\mathbf{X}) & \sigma_{32}(\mathbf{X}) & \sigma_3^2(\mathbf{X}) \end{pmatrix} \tag{16}$$

In the trivariate case, the estimated variance-covariance matrix can be non-positive-definite. Thus, $\hat{\Sigma}$ is modified by applying the unweighted bending method of Schaeffer (2014) as implemented in the `mbend` R package (Nilforooshan, 2020).

Following equation (15), a trivariate reference region may be estimated as:

$$R_\tau(\mathbf{X}) = \begin{pmatrix} \mu_1(\mathbf{X}) \\ \mu_2(\mathbf{X}) \\ \mu_3(\mathbf{X}) \end{pmatrix} + \Sigma^{1/2}(\mathbf{X})\varepsilon_\tau \tag{17}$$

where ε_τ is the unconditionally probabilistic region for the errors $(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ as

$$\varepsilon_\tau(k) = \{(\varepsilon_1, \varepsilon_2, \varepsilon_3) \in \mathbb{R}^3 | f(\varepsilon_1, \varepsilon_2, \varepsilon_3) \leq k\} \tag{18}$$

f being the density function of the trivariate residuals $(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ and k is the τ -quantile of $f(\varepsilon_1, \varepsilon_2, \varepsilon_3)$.

Using this model, the application of the methodology for diabetes diagnosis can be extended by incorporating an additional glycemia test. This extension is justified since other glyated proteins are routinely monitored in diabetes control besides FPG and HbA1c. For instance, in conditions that determine alterations in hemoglobin metabolism (e.g., anemia or kidney disease), fructosamine (Fr) is frequently used as an additional marker. Nevertheless, the translation of Fr into average glucose levels is not as clear as for HbA1c, and discordances are often encountered between the Fr and HbA1c results. In addition, agreement among these glycemia markers can be affected by factors such as patient age.

Figure 5 show the FPG, HbA1c and Fr results for the AEGIS sample. As can be seen, the values recorded for these tests correlate with one another, showing a complex multivariate distribution. Moreover, it can be appreciated how their multivariate distribution changes with age. To estimate a trivariate reference region for these markers, taking into account patient age, the `trivRegr()` function was used. This function is an extension of `bivRegr()` to the trivariate setting. The usage of both functions is similar, but additional additive predictors must be defined for the means vector and variance-covariance matrix.

```
R> dm_no = subset(aegis, aegis$dm == "no")
R> mu1 = fpg ~ s(age)
R> mu2 = hba1c ~ s(age)
R> mu3 = fru ~ s(age)

R> var1 = ~ s(age)
R> var2 = ~ s(age)
R> var3 = ~ s(age)

R> rho12 = ~ s(age)
```

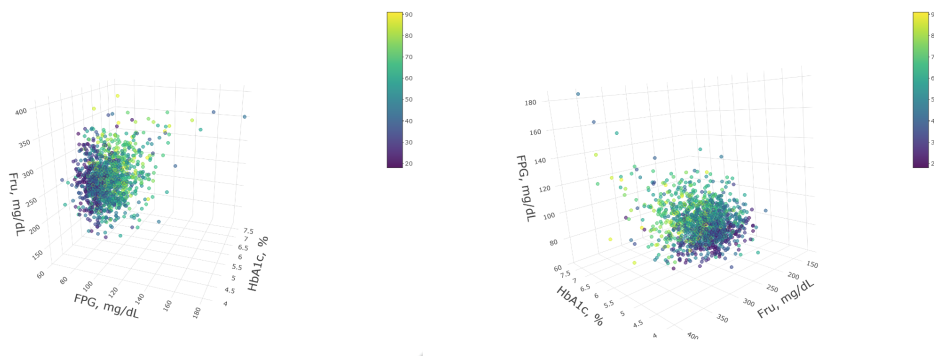


Figure 5: Scatter plot for three glyceic markers (fasting plasma glucose, glyated hemoglobin and fructosamine), with colour scale depending on age. The joint values of these glyceic markers seems to change with age, i.e., higher values are observed at higher ages. A trivariate and age-dependent reference region is desirable.

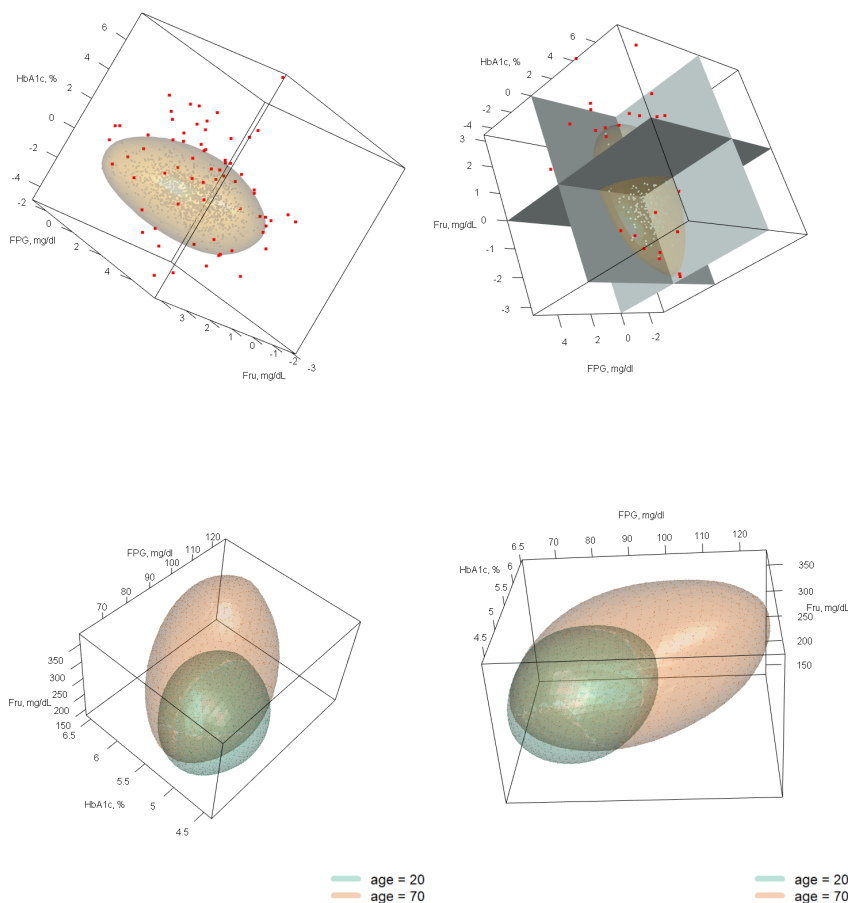


Figure 6: Standardized reference region (toprow plots), and conditional reference region (bottomrow plots), for three glyceic tests. Red points represent the trivariate values located outside the reference region after adjusting by age. The grey panels define eight octanes – each one with a different clinical profile. The trivariate reference region for these markers changes with age.

```
R> rho13 = ~ s(age)
R> rho23 = ~ s(age)

R> formula = list(mu1,mu2,mu3,var1,var2,var3,rho12,rho13,rho23)
R> fit = trivRegr(formula,data=dm\_no)
```

As in the bivariate estimation, the `trivRegion` function is applied to a `trivRegr` object. Here, the method was implemented only for a single τ (the kernel density bandwidth selection was not formally tested). The `plot()` method implemented for the `trivRegion` object can be used to interactively check the trivariate standardized and conditional reference regions:

```
R> region = trivRegion(fit,tau=0.95)

R> plot(region,planes = F,size=5, col="red", incol = "grey",xlab="FPG, mg/dl",
        ylab="HbA1c, %", zlab="Fru, mg/dL")
R> plot(region,planes = T,size=5,col="red",incol = "grey",xlab="FPG, mg/dl",
        ylab="HbA1c, %", zlab="Fru, mg/dL")
R> plot(region,cond=T,newdata=data.frame(age=c(20,70)), xlab="FPG, mg/dl",
        ylab="HbA1c, %", zlab="Fru, mg/dL", legend=T)
```

In Figure 6 the trivariate standardized and conditional reference regions are shown for different angles. The model residuals are centered around zero, with variance zero, and zero linear correlation. The region contains the 94.96% of the patients. In the trivariate setting, a patient may be located outside the region for different reasons. Indeed, if `plot()` argument `panels=T`, eight reasons exist for why a patient is located outside the reference region (explaining each situation goes beyond the scope of the present work). The conditional reference region may be produced for different ages by setting the `cond = T`, and providing new age values in `newdata`.

Case 2: beyond the medical research – Joint prediction of SO₂ and NO_x pollutants

This section illustrates how `refreg` methodology can be used in fields other than laboratory medicine. It is here shown how an uncertainty region useful for the joint forecasting of the concentration of two air pollutants (SO_2 and NO_x) can be derived using the `bivRegr` and `bivRegion` functions. The following example estimates which joint SO_2 and NO_x values are more likely in the course of a pollution episode. The data, which are contained in the package, were obtained from the surroundings of the a coal-fire power station in the northern Spain. Current Spanish legislation places a limit on the mean of 24 successive determinations of pollution concentration taken at 5 minute intervals in the neighborhood of potential point sources of pollution. Thus, access was available to historical concentrations of both air pollutants over a year, as well as several records of pollution episodes.

Given the historical concentrations of both pollutants (contained in the `pollution` dataset), we aimed to predict the SO_2 and NO_x concentrations during a specific pollution episode (contained in the `pollution_episode` dataset) 30 minutes in advance. As can be seen in the following R output, both datasets have a similar structure, SO_2 and NO_x are the current concentrations of both pollutants, while the remaining columns represent their concentrations in the previous 30, 45, and 60 minutes.

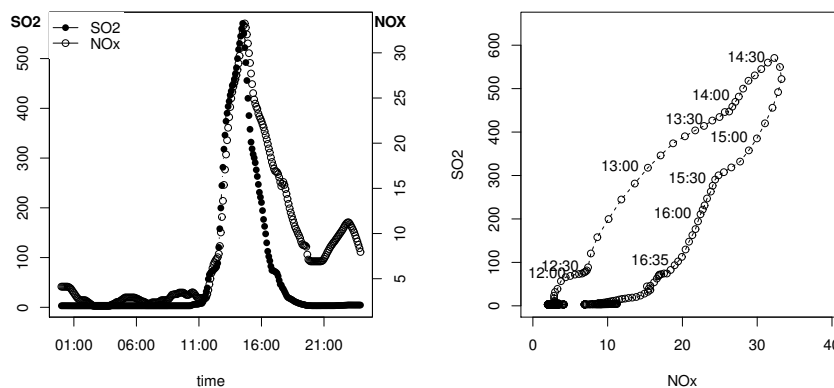


Figure 7: Two different representations of a pollution incident for SO_2 and NO_x . Left plot represent the pollution episode in the univariate scale, and right plot in a bivariate scale. SO_2 and NO_x pollution episodes are associated.

```
R > head(pollution[,1:9])
      Date      So2  Nox  So2_0  Nox_0  So2_1  Nox_1  So2_2  Nox_2
316 2003-02-07 16:15:00 38.38  2.38  73.50  3.21  76.79  3.38  81.92  3.54
1865 2003-05-07 03:10:00  3.00  3.67   3.00  3.33   3.00  3.33   3.00  3.33
1383 2003-04-04 01:35:00 256.50 11.17 293.71  8.96 294.54  8.33 285.29  7.71
```

```

3262 2003-07-09 14:35:00 225.29 17.04 104.67 7.67 84.33 6.29 64.67 4.67
1191 2003-03-30 13:55:00 42.33 4.12 80.21 4.96 84.38 5.00 88.33 5.00
3065 2003-07-04 11:50:00 145.83 8.58 99.12 5.71 83.83 5.25 70.92 4.83
    
```

```

R > head(pollution_episode[,1:9])
      Date So2 Nox So2_0 Nox_0 So2_1 Nox_1 So2_2 Nox_2
1 2003-05-09 00:00:00 3.08 4.12 3.08 4.50 3.08 4.46 3.08 4.38
2 2003-05-09 00:05:00 3.08 4.12 3.08 4.46 3.08 4.50 3.08 4.46
3 2003-05-09 00:10:00 3.08 4.12 3.08 4.38 3.08 4.46 3.08 4.50
4 2003-05-09 00:15:00 3.08 4.12 3.08 4.29 3.08 4.38 3.08 4.46
5 2003-05-09 00:20:00 3.08 4.12 3.08 4.21 3.08 4.29 3.08 4.38
6 2003-05-09 00:25:00 3.08 4.12 3.08 4.12 3.08 4.21 3.08 4.29
    
```

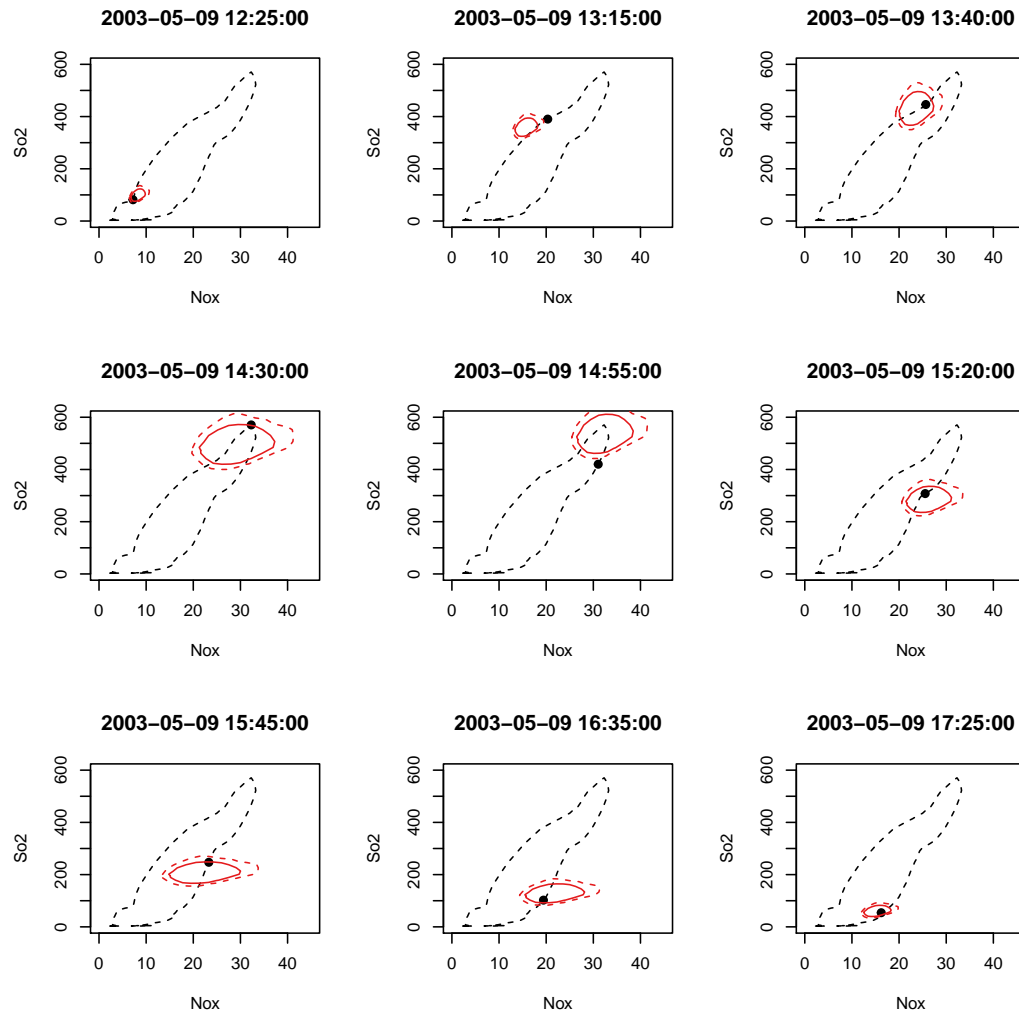


Figure 8: Estimated bivariate uncertainty region for a pollution episode. Black dashed line represents the pollution episode evolution, solid black point the observed value, and red contours the predicted uncertainty region for $\tau = 0.95$ (red solid contour) and $\tau = 0.975$ (red dashed contour).

Figure 7 shows the course of the pollution episode under prediction. In the left plot the SO_2 and NO_x concentrations over time are represented by solid and open circles, respectively. Each point in the right plot of this figure shows the concentration of both pollutants at a specific instant in time. It can be seen how, during a pollution episode, the concentration of both pollutants increases to a peak, and then returns slowly back to lower values. NO_x increases in a manner similar to the SO_2 , but its decrease is slower. Both representations show an evident correlation between the concentration of these air pollutants. To monitor this pollution episode, the historical records from the power plant were used, and NO_x levels made dependent on their previous values (30, and 60, minutes before; NO_{x_0} , and NO_{x_2}). Similarly, the SO_2 concentration was made dependent on its prior concentration

(So_{2_0} , So_{2_2}). Finally, the correlation between both was made dependent on the previous observation for both air pollutants (Nox_0 , So_{2_0}):

```
R> mu1 = Nox~s(Nox_0)+s(Nox_2)
R> mu2 = So2~s(So2_0)+s(So2_2)
R> var1 = ~s(Nox_0)+s(Nox_2)
R> var2 = ~s(So2_0)+s(So2_2)
R> rho = ~s(Nox_0)+s(So2_0)
R> f = list(mu1,mu2,var1,var2,rho)

R> fit = bivRegr(f,data=pollution)
R> region = bivRegion(fit,tau=c(0.950,0.975),shape=10)
```

In the previous code example, the reference region was estimated for $\tau = 0.95, 0.975$ for the model-standardized residuals. A pollution episode can then be forecasted by predicting the uncertainty regions based on the values provided by the `pollution_episode` dataset. Note, that one dataset is used to fit the model, and another when making predictions. The observed SO_2 and NO_x concentrations during the pollution episode are shown in Figure 8 along with the predicted probabilistic regions. Each plot corresponds to a different instant in time, and to different pollution episode states. The upper right-hand side shows the beginning of the pollution episode, the bottom left plot shows its ending. The region's shape changes over time, anticipating reasonably well the evolution of the pollution episode. The size of the region becomes larger as the pollution peak is approached, and then gradually becomes smaller. This is to be expected since the maximum corresponds to a transition between the increase and decrease of the concentrations of both pollutants, a situation that involves more uncertainty. Moreover, at the end of the pollution episode, the region is higher on the X axis, which corresponds to greater uncertainty for the Nox prediction. As commented above, this might be explained in that the NO_x concentration decreases more slowly than the SO_2 concentration.

```
par(mfrow=c(3,3))
for(k in c(150, 160, 165, 175, 180, 185, 190, 200,210)){
  plot(pollution_episode[,3:2],type="l",lty=2,ylim=c(0,600),xlim=c(0,45),
  main=pollution_episode[k,1])
  points(pollution_episode[k,3:2],col="black",pch=19)
  plot(region, cond = T, newdata = pollution_episode[k,], add = T,
  tau=c(0.95, 0.975),legend=F)
}
```

5 Concluding remarks

This paper discusses the R implementation of a newly developed method for estimating conditional reference regions. The method was originally designed for bivariate responses to provide a joint interpretation of two glycemia markers. However, as shown with real data, the proposed package can be used in other fields, and its extension to three dimensions is feasible. The proposed package is useful in the definition of conditional reference regions for continuous diagnostic tests. Few MVRs applications are used in practice, yet they have been shown clinically valuable in the treatment of patients with cancer (Mattsson et al., 2008), cardiovascular disease (Selmerud et al., 2018) and endocrine problems (Hoermann et al., 2016). Given the simplicity with which `refreg` can be used, and its having no parametric restrictions, it is hoped it might enhance the use of MVRs.

The definition of a region characterizing the central part of a multivariate distribution may be of great interest in other fields. For instance, in quality control studies, multivariate control charts are commonly used when two or more attributes of a product, or process, require evaluation (Fuchs and Kenett, 1998). Analogously to medical MVRs, this multivariate analysis is currently performed assuming a Gaussian distribution. Thus, the `refreg` R package may help provide for better quality surveillance of medical conditions, environmental problems and even industrial processes that are monitored by the measurement of more than one variable.

Future versions of our package will look forward to extending our model for continuous responses of dimension higher than three. Such reference region would require a kernel density estimator for high dimension applications (Nagler, 2021). To the best of our knowledge, the estimated region is not easy to visualize for scales larger than three. Although, we might identify and visualize the multivariate values located inside or outside the region using parallel coordinate plots, or the interactive visualization methods as the implemented in the `tourr` package (Wickham et al., 2011).

6 Acknowledgements

Óscar Lado-Baleato is funded by a pre-doctoral grant (ED481A-2018) from the Galician Government (Plan I2C)-Xunta de Galicia. This study was also supported by grants from the Carlos III Institute of Health (Instituto de Salud Carlos III-ISCIII/PI20/01069/Co-funded by European Union), the Network for Research on Chronicity, Primary Care, and Health Promotion (Instituto de Salud Carlos III-ISCIII/RD21/0016/0022/Co-funded by European Union), and the Galician Innovation Agency-Competitive Benchmark Groups (GAIN-GRC/IN607A/2021/02/Xunta de Galicia). This article was developed under the project MTM2017-83513-R, cofinanced by the Ministry of Economy and Competitiveness (SPAIN) and by the European Regional Development Fund (FEDER). The work was also supported by the project ED431C 2020/20, approved within the framework of the Competitive Research Unit Consolidation Programme (2020), Xunta de Galicia. Work supported by the Grant PID2020-118101GB-I00, Ministerio de Ciencia e Innovación (MCIN/ AEI /10.13039/501100011033).

Bibliography

- H. Aleyassine, R. Gardiner, D. Tonks, and P. Koch. Glycosylated hemoglobin in diabetes mellitus: Correlations with fasting plasma glucose, serum lipids and glycosuria. *Diabetes Care.*, 3(4):508–514, 1980. [p140]
- American Diabetes Association Classification and diagnosis of diabetes: Standards of medical care in diabetes-2019. *Diabetes Care.*, 42(Suppl 1):13–27, 2019. [p140]
- J. C. Boyd. Reference regions of two or more dimensions. *Clinical Chemistry and Laboratory Medicine (CCLM)*, 42(7):739–746, 2004. [p135]
- T. J. Cole and P. J. Green. Smoothing reference centile curves: the lms method and penalized likelihood. *Stat Med.*, 11(10):1305–1319, 1992. [p135]
- M. B. Davidson. The effect of aging on carbohydrate metabolism: a review of the english literature and a practical approach to the diagnosis of diabetes mellitus in the elderly. *Metab Clin Exp.*, 28(6): 688–705, 1979. [p140]
- J. Espasandín-Domínguez, C. Cadarso-Suárez, T. Kneib, G. Marra, N. Klein, R. Radice, O. Lado-Baleato, A. González-Quintela, and F. Gude. Assessing the relationship between markers of glycemic control through flexible copula regression models. *Statistics in medicine*, 38(27):5161–5181, 2019. [p135]
- C. Fuchs and R. S. Kenett. *Multivariate quality control: theory and applications*. CRC Press, 1998. [p148]
- F. Gude, P. Díaz-Vidal, C. Rúa-Pérez, M. Alonso-Sampedro, C. Fernández-Merino, J. Rey-García, C. Cadarso-Suárez, M. Pazos-Couselo, J. M. García-López, and A. Gonzalez-Quintela. Glycemic variability and its association with demographics and lifestyles in a general adult population. *J Diabetes Sci Technol.*, 11(4):780–790, 2017. [p140]
- M. J. Hallworth. The ‘70% claim’: what is the evidence base? *Ann Clin Biochem.*, 48(6):487–488, 2011. [p135]
- R. Hoermann, R. Larisch, J. W. Dietrich, and J. E. Midgley. Derivation of a multivariate reference range for pituitary thyrotropin and thyroid hormones: diagnostic efficiency compared with conventional single-reference method. *European journal of endocrinology*, 174(6):735–743, 2016. [p148]
- Z. Hu and R. cai Yang. *distfree.cr: Distribution-Free Confidence Region*, 2018. URL <https://CRAN.R-project.org/package=distfree.cr>. R package version 1.5.1. [p136]
- R. J. Hyndman. *hdrcde: Highest Density Regions and Conditional Density Estimation*, 2018. URL <http://pkg.robjhyndman.com/hdrcde>. R package version 3.3. [p136]
- M. K. Kim, J. S. Jeong, J.-S. Yun, H.-S. Kwon, K. H. Baek, K.-H. Song, Y.-B. Ahn, and S.-H. Ko. Hemoglobin glycation index predicts cardiovascular disease in people with type 2 diabetes mellitus: A 10-year longitudinal cohort study. *J Diabetes Complicat.*, 32(10):906–910, 2018. [p140]
- R. Koenker and G. Bassett Jr. Regression quantiles. *Econometrica.*, 46(1):33–50, 1978. [p135]
- O. Lado-Baleato, Roca-Pardiñas, C. Cadarso-Suárez, and F. Gude. Modelling conditional reference regions. application to glycemic markers. *Stat Med.*, 2021. Under Review. [p144]

- A. Magnusson and J. Burgos. *r2d2: Bivariate (Two-Dimensional) Confidence Region and Frequency Distribution*, 2014. URL <https://CRAN.R-project.org/package=r2d2>. R package version 1.0-0. [p136]
- A. Mattsson, D. Svensson, B. Schuett, K. J. Osterziel, and M. B. Ranke. Multidimensional reference regions for igf-i, igfbp-2 and igfbp-3 concentrations in serum of healthy adults. *Growth Hormone & IGF Research*, 18(6):506–516, 2008. [p148]
- Microsoft 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. [p142]
- Microsoft and S. Weston. *foreach: Provides Foreach Looping Construct*, 2020. URL <https://CRAN.R-project.org/package=foreach>. R package version 1.5.1. [p142]
- T. Nagler. *kdevine: Multivariate Kernel Density Estimation with Vine Copulas*, 2021. URL <https://CRAN.R-project.org/package=kdevine>. R package version 0.4.3. [p148]
- M. A. Nilforooshan. mbend: an R package for bending non-positive-definite symmetric matrices to positive-definite. *BMC genetics*, 21(1):1–8, 2020. [p144]
- L. N. Pani, L. Korenda, J. B. Meigs, C. Driver, S. Chamany, C. S. Fox, L. Sullivan, R. B. D'Agostino, and D. M. Nathan. Effect of aging on a1c levels in individuals without diabetes: evidence from the framingham offspring study and the national health and nutrition examination survey 2001–2004. *Diabetes Care.*, 31(10):1991–1996, 2008. [p140]
- J. Roca-Pardiñas, C. Ordóñez, and O. Lado-Baleato. Nonparametric location–scale model for the joint forecasting of so₂ and no_x pollution episodes. *Stochastic Environmental Research and Risk Assessment*, pages 1–14, 2020. [p136, 138]
- L. Schaeffer. Making covariance matrices positive definite. *Center for Genetic Improvement of Livestock*. <http://animalbiosciences.uoguelph.ca/~lrs/ELARES/PDforce.pdf> (Accessed 1 September 2017), 2014. [p144]
- J. Selmer, E. Henriksen, H. Dalen, and P. Hedberg. Derivation and evaluation of age-specific multivariate reference regions to aid in identification of abnormal filling patterns: the hunt and vamis studies. *JACC: Cardiovascular Imaging*, 11(3):400–408, 2018. [p148]
- D. M. Stasinopoulos, R. A. Rigby, et al. Generalized additive models for location scale and shape (gamlss) in r. *Journal of Statistical Software*, 23(7):1–46, 2007. [p135]
- M. D. Stasinopoulos, R. A. Rigby, G. Z. Heller, V. Voudouris, and F. De Bastiani. *Flexible regression and smoothing: using GAMLSS in R*. CRC Press, 2017. [p135]
- Y. Wei. An approach to multivariate covariate-dependent quantile contours with application to bivariate conditional growth charts. *J Am Stat Assoc.*, 103(481):397–409, 2008. [p135]
- WHO. *Multicentre Growth Reference Study Group*. World Health Organization, 2006. [p135]
- H. Wickham, D. Cook, H. Hofmann, and A. Buja. *tourr: An R package for exploring multivariate data with projections*. *Journal of Statistical Software*, 40:1–18, 2011. [p148]
- P. Winkel and J. Lyngbye. The normal region—a multivariate problem. *Scandinavian journal of clinical and laboratory investigation*, 30(3):339–344, 1972. [p135]
- C. Winkler, K. Linden, A. Mayr, T. Schultz, T. Welchowski, J. Breuer, and U. Herberg. Refcurv: A software for the construction of pediatric reference curves. *arXiv preprint arXiv:1901.09775*, 2019. [p135]
- S. N. Wood. *Generalized additive models: an introduction with R*. CRC press, 2017. [p138]
- E. M. Wright and P. Royston. Calculating reference intervals for laboratory measurements. *Stat Method Med Res.*, 8(2):93–112, 1999. [p135]
- D. S. Young. tolerance: An r package for estimating tolerance intervals. *Journal of Statistical Software*, 36(i05), 2010. [p136]
- M. Šiman and P. Boček. *modQR: Multiple-Output Directional Quantile Regression*, 2019. URL <https://CRAN.R-project.org/package=modQR>. R package version 0.1.2. [p136]

Óscar Lado-Baleato

Department of Statistics, Mathematical Analysis, and Optimization, Universidade de Santiago de Compostela, Galicia, Spain.

Medicine Colleague, Rúa San Francisco, Santiago de Compostela.

Galicia, Spain

(<https://orcid.org/0000-0001-9592-4879>)

oscarlado.baleato@usc.es

Javier Roca-Pardiñas

Galician Center for Mathematical Research and Technology (CITMAga) & Statistical Inference, Decision and Operations Research, Universidade de Vigo.

Rúa do Conde de Torrecedeira, 86, 36310 Vigo, Pontevedra.

Galicia, Spain

roca@uvigo.com

Carmen Cadarso-Suárez

Galician Center for Mathematical Research and Technology (CITMAga) & Department of Statistics, Mathematical Analysis, and Optimization, Universidade de Santiago de Compostela, Galicia, Spain.

Medicine Colleague, Rúa San Francisco, Santiago de Compostela.

Galicia, Spain

carmen.cadarso@usc.es

Francisco Gude

Clinical Epidemiology Unit, Complexo Hospitalario de Santiago de Compostela.

Rúa da Choupana, s/n, 15706 Santiago de Compostela, A Coruña

Galicia, Spain

francisco.gude.sampedro@sergas.es

TensorTest2D: Fitting Generalized Linear Models with Matrix Covariates

by Ping-Yang Chen, Hsing-Ming Chang, Yu-Ting Chen, Jung-Ying Tzeng, and Sheng-Mao Chang

Abstract The **TensorTest2D** package provides the means to fit generalized linear models on second-order tensor type data. Functions within this package can be used for parameter estimation (e.g., estimating regression coefficients and their standard deviations) and hypothesis testing. We use two examples to illustrate the utility of our package in analyzing data from different disciplines. In the first example, a tensor regression model is used to study the effect of multi-omics predictors on a continuous outcome variable which is associated with drug sensitivity. In the second example, we draw a subset of the MNIST handwritten images and fit to them a logistic tensor regression model. A significance test characterizes the image pattern that tells the difference between two handwritten digits. We also provide a function to visualize the areas as effective classifiers based on a tensor regression model. The visualization tool can also be used together with other variable selection techniques, such as the LASSO, to inform the selection results.

1 Introduction

Tensors are multidimensional arrays and are increasingly encountered in practices due to the burgeoning development of high throughput technology, e.g., brain image data (Zhou et al., 2013) and multi-omics data (Chang et al., 2021). Within the framework of regression analysis, tensor-structured data can play a role in the response variable, the explanatory variable, or in both. Some available R packages, such as **TRES** and **MultiwayRegression**, consider tensor regression with general tensor structure. The package **TRES** (Wang et al., 2020) provides tools to perform regression analysis with a tensor envelope structure in the tensor regression model, and the output of which includes p -values for the regression coefficients. **TRES** aims at variable selection via significance tests. The package **MultiwayRegression** (Lock, 2018, 2019) performs L_2 penalized tensor regression which is useful for predictive modeling but not for the identification of important variables. Both the **TRES** and the **MultiwayRegression** consider regression models with continuous outcome variables only. Compared to existing R packages, the proposed package **TensorTest2D** (Chen et al., 2021) considers a generalized linear model (GLM) with matrix-structured predictors and a scalar outcome, and it can be used for outcome prediction or testing.

There are four main functions in **TensorTest2D**. The function `tensorReg2D()` is designed to provide estimates of regression coefficients and their standard deviations, as well as the p -value for testing whether a regression coefficient is significantly different from zero. The function `summary()` organizes the above information into an output table. The function `plot()` can be used to visualize the locations of the predictors significantly affecting the response variable in the predictor matrix. Finally, the function `predict()` can be used to predict the response values using the conditional mean given a specific predictor matrix.

The rest of this paper is arranged as follows. First, we describe a regression model with tensor predictors under GLM. Next, we illustrate the main functions in package **TensorTest2D** using two examples, and illustrate the relevancy of using low-rank tensor regression. The first example focuses on association testing, where we apply tensor regression to identify genomic variables that affect the drug sensitivity for lung cancer treatment. The second example is for classification, where we apply logistic tensor regression to model the relationship between a binary response variable and images of handwritten digits in the MNIST database. We also use significance testing of the image predictors to identify locations of an image that play important roles in distinguishing between two different handwritten digits. The datasets being used in these two examples are included in the package **TensorTest2D** — users can use `data(omics)` to load the dataset of the first (association) example and `data(mnist_mp2c2)` to load the dataset of the second (classification) example.

2 Generalized tensor regression model

In this work, we consider tensor regression under the GLM framework and extend the inference procedure of tensor parameters in Chang et al. (2021) from continuous responses to binary and count responses. Suppose that there exists a dataset consisting of n independent triplets, $\{y_i, \mathbf{w}_i, \mathbf{X}_i\}$, $i = 1, \dots, n$, where y_i is a scalar outcome, \mathbf{w}_i is a d -dimensional covariate vector and \mathbf{X}_i is a $P \times Q$ matrix predictor. Without loss of generality, we assume hereafter $P \leq Q$. For two matrices, say \mathbf{X}_1 and

\mathbf{X}_2 , of the same size, define the dot product of these two matrices as $\mathbf{X}_1 \circ \mathbf{X}_2 = \sum_{p=1}^P \sum_{q=1}^Q X_{1,pq} X_{2,pq}$ where $X_{j,pq}$ is the (p, q) th entry of the matrix $\mathbf{X}_j, j = 1, 2$. The GLM with an order-2 tensor predictor can therefore be defined as

$$g(E(y_i)) = \mathbf{w}_i^\top \boldsymbol{\beta} + \mathbf{X}_i \circ \mathbf{B}, \tag{1}$$

where $g(\cdot)$ is the link function, $\boldsymbol{\beta} \in \mathbb{R}^d$ and $\mathbf{B} \in \mathbb{R}^{P \times Q}$. If there are $P \times Q$ unconstrained parameters in \mathbf{B} , the above representation is equivalent to a glm with $d + PQ$ covariates, including the intercept. In **TensorTest2D**, we implement linear regression with identity link, Poisson regression with log link, and logistic regression with logit link.

When PQ is relatively small, one can vectorize the matrix \mathbf{X}_i so that (1) can be expressed as a conventional GLM with $d + PQ$ covariates. When PQ is large, one can explore the matrix structure of predictors and consider a low-rank tensor GLM so as to reduce the number of parameters of interest while retaining the variable-specific resolution. The main idea of tensor GLM is to model \mathbf{B} by a low-rank-constrained \mathbf{B} so that \mathbf{B} can be fully specified using fewer unconstrained parameters. See [Hung and Wang \(2012\)](#) and [Chang et al. \(2021\)](#) for more detail. Take the MNIST handwritten image classification as an example, where handwritten images are recorded in 10×10 matrices. When there is no constraint on \mathbf{B} , the number of parameters of interest is $PQ = 100$. On the other hand, if we restrict the rank of \mathbf{B} to be r , the number of unconstrained parameters is $(P + Q) \times r - r^2$ that takes a value of 19, 36, 51, and 64 when $r = 1, 2, 3,$ and 4 , respectively. The reduction in the number of parameters is significant when r is small.

Given a pre-specified rank $r = R$, one can model \mathbf{B} with a low-rank constraint by setting $\mathbf{B} = \mathbf{B}_1 \mathbf{B}_2^\top$, where $\mathbf{B}_1 \in \mathbb{R}^{P \times R}, \mathbf{B}_2 \in \mathbb{R}^{Q \times R}$ and $R \leq P$ ([Zhou et al., 2013](#); [Chang et al., 2021](#)). The low-rank tensor regression model is therefore

$$g(E(y_i)) = \mathbf{w}_i^\top \boldsymbol{\beta} + \mathbf{X}_i \circ (\mathbf{B}_1 \mathbf{B}_2^\top). \tag{2}$$

Additional constraints on $\mathbf{B}_1 \mathbf{B}_2^\top$ are needed to ensure parameter identifiability, see [Zhou et al. \(2013\)](#) for example. We adopt in this package the constraints considered by [Chang et al. \(2021\)](#), that leaves the total number of unconstrained parameters to be $d + (P + Q)R - R^2$. Denote $\boldsymbol{\eta}$ as the vector which collects all unconstrained parameters in (2). According to the theory of GLM, the score function and the Fisher information matrix with respect to the model are

$$\sum_{i=1}^n \frac{\partial \mu_i}{\partial \boldsymbol{\eta}} (y_i - \mu_i) \quad \text{and} \quad \sum_{i=1}^n \frac{\partial \mu_i}{\partial \boldsymbol{\eta}} V_i \left(\frac{\partial \mu_i}{\partial \boldsymbol{\eta}} \right)^\top,$$

respectively, where $\mu_i = E(y_i)$ and $V_i = \text{Var}(y_i)$. However, we are interested in estimating $\mathbf{B} = \mathbf{B}_1 \mathbf{B}_2^\top$ and testing whether entries of \mathbf{B} are nonzero. The derivation of the sampling distribution of $\hat{\mathbf{B}}$ is omitted here, for the process is similar to that in [Chang et al. \(2021\)](#) and it does not need to be reproduced here again to misdirect readers' attention. As the true rank of \mathbf{B} is unknown, following [Chang et al. \(2021\)](#), we use the Akaike information criterion (AIC) to determine the optimal rank.

Before giving a brief description of how [Chang et al. \(2021\)](#) place constraints on tensor regression parameterization, we wish to emphasize that the process of estimating for $\partial \mu_i / \partial \boldsymbol{\eta}$ is in typical not exactly simple. It is known that the matrix factorization (decomposition) $\mathbf{B} = \mathbf{B}_1 \mathbf{B}_2^\top$ is not unique because, for every invertible matrix $\mathbf{O} \in \mathbb{R}^{R \times R}, \mathbf{B} = (\mathbf{B}_1 \mathbf{O}^{-1}) (\mathbf{O} \mathbf{B}_2^\top)$. Write

$$\mathbf{B}_1 = \begin{bmatrix} \mathbf{B}_{11} \\ \mathbf{B}_{21} \end{bmatrix},$$

where $\mathbf{B}_{11} \in \mathbb{R}^{R \times R}$ and $\mathbf{B}_{21} \in \mathbb{R}^{(P-R) \times R}$, and assume \mathbf{B}_{11} is invertible. One way to ensure the uniqueness of the matrix factorization is to force $\mathbf{O} = \mathbf{B}_{11}$, and thus

$$\mathbf{B}_1 \mathbf{B}_2^\top = \begin{bmatrix} \mathbf{B}_{11} \\ \mathbf{B}_{21} \end{bmatrix} \mathbf{O}^{-1} \mathbf{O} \mathbf{B}_2^\top = \begin{bmatrix} \mathbf{B}_{11} \mathbf{O}^{-1} \\ \mathbf{B}_{21} \mathbf{O}^{-1} \end{bmatrix} \tilde{\mathbf{B}}_2^\top = \begin{bmatrix} \mathbf{I}_R \\ \tilde{\mathbf{B}}_{21} \end{bmatrix} \tilde{\mathbf{B}}_2^\top,$$

where $\tilde{\mathbf{B}}_{21} = \mathbf{B}_{21} \mathbf{O}^{-1}$ and $\tilde{\mathbf{B}}_2 = \mathbf{B}_2 \mathbf{O}^\top$. Consequently, the unknown parameter matrices are $\tilde{\mathbf{B}}_{21} \in \mathbb{R}^{(P-R) \times R}$ and $\tilde{\mathbf{B}}_2 \in \mathbb{R}^{Q \times R}$ with a total of $(P - R) \times R + Q \times R$ unknown parameters. We believe that an exact formula for $\partial \mu_i / \partial \tilde{\boldsymbol{\eta}}$ can not be found prior to the work by [Chang et al. \(2021\)](#) in the case when $\tilde{\boldsymbol{\eta}} = (\text{vec}(\tilde{\mathbf{B}}_{12})^\top, \text{vec}(\tilde{\mathbf{B}}_1)^\top)^\top$, and the same formula is used in **TensorTest2D**.

3 Data analysis examples

In this section, we present examples of real data analysis by using the package `TensorTest2D`. The main function, `tensorReg2D`, in our `TensorTest2D` package is used for following data analysis. The inputs are the response vector, y , covariates matrix X , collecting tensor data, and vector W , collecting adjustment information such as age and gender. The key configurable parameters are the rank of B , n_R , and the type of response variable, `family`. The `tensorReg2D` handles three types of generalized regression problems. For continuous response, set `family = "gaussian"` and it fits the linear regression model based on identity link function. If the responses are binary, by setting `family = "binomial"`, it runs logistic regression modeling through the logit link. When the response variable is non-negative integer, the log link corresponding to poisson regression is used by setting `family = "poisson"`.

The function `tensorReg2D()` returns a list object which includes the following variables: `b_EST` represents the coefficient vector $\hat{\beta}$; `b_SD` represents the corresponding standard deviation vector and `b_PV` the p -value vector; `B_EST` represents the coefficient matrix \hat{B} for the image effect, `B_SD` the standard deviations of the coefficient estimates and `B_PV` the matrices of p -values; the output `IC` contains the AIC and BIC values for the purpose of model selection.

See `?tensorReg2D` for more details of the configuration and the output values.

Example 1: Tensor regression for continuous response using CCLE dataset

The package `TensorTest2D` includes a data set, `omics`, which consists of a continuous response variable and 30 omics predictors that can be organized into a 3×10 matrix. The response variable is the drug sensitivity of vandetanib measured by log-transformed activity area. Vandetanib is a drug targeting gene EGFR for lung cancer treatment. The 30 omics predictors are the genomic information of 10 genes measured from 3 platforms: copy number variation (CNV), methylation and mRNA expression. Among the 10 genes, 7 of them (EGFR, EREG, HRAS, KRAS, PTPN11, STAT3, and TGFA) are involved in the protein-protein interaction network of EGFR (<https://string-db.org>) and the rest (ACTB, GAPDH, and PPIA) are arbitrarily chosen housekeeping genes with permuted entries and serve as negative controls. The included data, `omics.RData`, is a subset of the data set provided by cancer cell line encyclopedia (CCLE) project (Barretina et al. (2012); <https://sites.broadinstitute.org/ccle/>). Detailed pre-processing procedure for `omics` is available in (Chang et al., 2021). The data set `omics` can be loaded via the following syntax:

```
library(TensorTest2D)
data(omics)
# The size of the data P, Q, n
print(dim(omics$omics))

#> [1] 3 10 68
```

In the `omics` example, w_i only consists of intercepts and X_i being a $P \times Q$ matrix with $P = 3$ and $Q = 10$. As described, this matrix consists of expression values of 10 genes evaluated under three different platforms. For the reason of $R \leq \min\{P, Q\}$ (see Chang et al. (2021)), there are three possible tensor models, namely, the rank-1, the rank-2 and the rank-3 model, to describe the relationship between the outcome and the matrix predictors. The models with the smallest AIC value will be selected as the optimal one, and here the rank-1 model has the smallest AIC value. The rank-1 model identifies two important variables: EGFR under methylation platform (coefficient = -0.2416; p -value = 0.0022) and EGFR under CNV platform (coefficient = 0.2508; p -value = 0.0061). Those lines of code below this paragraph were used as an example to perform and print out the results of model fitting. The utility function `summary(omicsMdl)` shows the model structure, summary statistics about the residuals, and the table of significance tests for the coefficients. On top of the result table, the model structure $y \sim I + X$ of this case is revealed where I is the intercept term and X is the matrix covariate. The names of the coefficients appear in the first column of the table. In addition to the (Intercept) and the terms of w , $X_{i \cdot j}$ is the coefficient of the i th row and j th column of X . If the row and the column names of X are specified, then the names of coefficients in X are `ROWi : COLUMNj`. Those values in the summary table can also be returned separately. Here, we print the attributes of the `tsglm` object separately for the estimated coefficients and their standard deviations, along with the p -values by the Wald test (see Wald (1943)).

```
set.seed(100) # Set seed for reproducibility
# Try from rank-1 to rank-3 models
omicsAIC <- numeric(3)
for (k in 1:3) {
  # Temporary storage for the rank-k model for withdrawing its AIC value
```



```

omicsTmp <- tensorReg2D(y = omics$Y, X = omics$omics,
                        W = matrix(1, length(omics$Y), 1),
                        n_R = k, family = "gaussian",
                        opt = 1, max_ite = 1000, tol = 10^(-7) )
omicsAIC[k] <- omicsTmp$IC[1] # AIC
}
sprintf('Rank-%d model is the best with smallest AIC = %4.4f', which.min(omicsAIC), min(omicsAIC))

#> [1] "Rank-1 model is the best with smallest AIC = -62.3135"

# Train the tensor regression model of rank 1
omicsMdl <- tensorReg2D(y = omics$Y, X = omics$omics,
                        W = matrix(1, length(omics$Y), 1),
                        n_R = which.min(omicsAIC), family = "gaussian",
                        opt = 1, max_ite = 1000, tol = 10^(-7) )

# Return the results of significance tests for all coefficients
summary(omicsMdl)

#> Call:
#> formula = y ~ I + X
#>
#> Residuals:
#>    Min.   1st Qu.   Median     Mean   3rd Qu.    Max.
#> -1.31835 -0.29160  0.03354  0.00000  0.40356  1.06511
#>
#> Coefficients:
#>
#> (Intercept)      Estimate Std. Error  t value Pr(>|t|)
#> cnv:EGFR         0.25078    0.08796  2.85098 0.0061255 ***
#> meth:EGFR        -0.24162    0.07511 -3.21673 0.0021740 ***
#> rna.rpkm:EGFR     0.08933    0.07857  1.13696 0.2604852
#> cnv:EREG         -0.00751    0.05094 -0.14743 0.8833301
#> meth:EREG         0.00724    0.0494  0.14648 0.8840812
#> rna.rpkm:EREG    -0.00268    0.01849 -0.14468 0.8854906
#> cnv:HRAS         -0.04866    0.05983 -0.81334 0.4195282
#> meth:HRAS         0.04689    0.06056  0.77425 0.4421012
#> rna.rpkm:HRAS    -0.01733    0.02956 -0.5865 0.5599385
#> cnv:KRAS         -0.0267    0.05067 -0.52699 0.6003203
#> meth:KRAS         0.02573    0.04913  0.52367 0.6026098
#> rna.rpkm:KRAS    -0.00951    0.01754 -0.54244 0.5897064
#> cnv:PTPN11       0.09193    0.06183  1.48669 0.1428065
#> meth:PTPN11      -0.08857    0.05093 -1.73886 0.0876539 *
#> rna.rpkm:PTPN11  0.03275    0.03289  0.99558 0.3238137
#> cnv:STAT3        -0.05747    0.05517 -1.0417 0.3021072
#> meth:STAT3        0.05537    0.04953  1.11792 0.2684585
#> rna.rpkm:STAT3   -0.02047    0.0257 -0.79672 0.4290423
#> cnv:TGFA         0.05049    0.06361  0.79367 0.4307973
#> meth:TGFA        -0.04864    0.05903 -0.82399 0.4135018
#> rna.rpkm:TGFA    0.01798    0.02625  0.68526 0.4960557
#> cnv:ACTB         -0.03107    0.05212 -0.5961 0.5535539
#> meth:ACTB         0.02993    0.05128  0.58371 0.5618033
#> rna.rpkm:ACTB    -0.01107    0.02339 -0.47307 0.6380374
#> cnv:GAPDH        -0.04123    0.04963 -0.83059 0.4097953
#> meth:GAPDH        0.03972    0.04737  0.83856 0.4053450
#> rna.rpkm:GAPDH   -0.01469    0.02221 -0.66128 0.5111932
#> cnv:PPIA         -0.04299    0.06373 -0.67454 0.5027957
#> meth:PPIA         0.04142    0.05989  0.69153 0.4921444
#> rna.rpkm:PPIA    -0.01531    0.02711 -0.56477 0.5745259
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Estimated coefficients
print(round(omicsMdl$B_EST, 3))

```

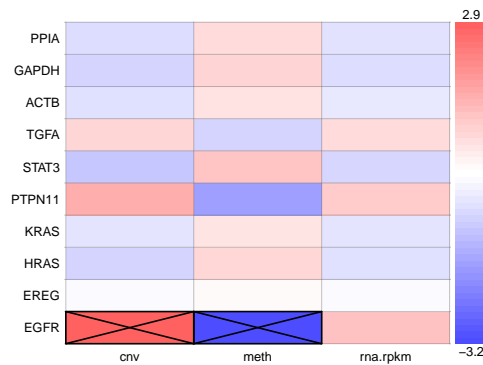


Figure 1: The image plot of the values for t-statistics of matrix covariate in the omics data. The effective pixels identified by the tensor regression model are marked out by the \boxtimes symbol.

```
#>          EGFR  EREG  HRAS  KRAS  PTPN11  STAT3  TGFA  ACTB  GAPDH  PPIA
#> cnv      0.251 -0.008 -0.049 -0.027  0.092 -0.057  0.050 -0.031 -0.041 -0.043
#> meth     -0.242  0.007  0.047  0.026 -0.089  0.055 -0.049  0.030  0.040  0.041
#> rna.rpkm  0.089 -0.003 -0.017 -0.010  0.033 -0.020  0.018 -0.011 -0.015 -0.015

# The standard deviation of the coefficients
print(round(omicsMdl$B_SD, 3))

#>          EGFR  EREG  HRAS  KRAS  PTPN11  STAT3  TGFA  ACTB  GAPDH  PPIA
#> cnv      0.088  0.051  0.060  0.051  0.062  0.055  0.064  0.052  0.050  0.064
#> meth     0.075  0.049  0.061  0.049  0.051  0.050  0.059  0.051  0.047  0.060
#> rna.rpkm  0.079  0.018  0.030  0.018  0.033  0.026  0.026  0.023  0.022  0.027

# The p-values of the coefficients by the Wald test
print(round(omicsMdl$B_PV, 3))

#>          EGFR  EREG  HRAS  KRAS  PTPN11  STAT3  TGFA  ACTB  GAPDH  PPIA
#> cnv      0.006  0.883  0.420  0.600  0.143  0.302  0.431  0.554  0.410  0.503
#> meth     0.002  0.884  0.442  0.603  0.088  0.268  0.414  0.562  0.405  0.492
#> rna.rpkm  0.260  0.885  0.560  0.590  0.324  0.429  0.496  0.638  0.511  0.575
```

In our package **TensorTest2D**, the function `plot()` can be used to visualize the importance of the matrix predictor. The output is an $P \times Q$ heat map that the plotted values on it are controlled by the option `type`. If the unit of data varies across the rows or columns in X , it is suggested to choose the t-statistics of the coefficients (`type = "tval"`) instead of their values `type = "coef"`. In addition, the function `plot()` also marks the pixels with p -values smaller than a pre-determined significance level. Users can select the p -value adjusting method (see `help("p.adjust")`) by the option `method` and specify the significance level through the option `alpha`. We plot in Figure 1 the t-statistics of the coefficients in $\hat{\mathbf{B}}$, where red and blue colors represent the pixels of positive and negative values, respectively. For those coefficients with p -values less than α , their corresponding pixels are marked with the symbol, \boxtimes , in this example, `cnv:EGFR` and `meth:EGFR`.

```
plot(x = omicsMdl, method = "none", alpha = 0.05, type = "tval",
     showlabels = TRUE, plot.legend = TRUE)
```

Example 2: Logistic tensor regression and classification using MNIST dataset

MNIST is a well-known benchmark database for image recognition in machine learning. It consists of over 60,000 training images and 10,000 testing images. For R users, one can obtain the image data from the **dslabs** package (Irizarry and Gill, 2019). For the purpose of demonstration, we reduce the size of MNIST image data by a max-pooling step with 2×2 block, as shown in the images on the left and in the middle of Figure 2. The original 28×28 images are thus pixelated into 14×14 max-pooled images. Because pixels at the edges and the corners of the max-pooled images take no information across almost all images, we removed those pixels and ended up with $P \times Q = 10 \times 10$ sub-images as illustrated in the image on the right of Figure 2. The mean image of the pre-processed training set for each label in the MNIST database is shown in Figure 3. These pre-processed images of 10×10 pixels are included in the package **TensorTest2D** and can be imported using the following commands:

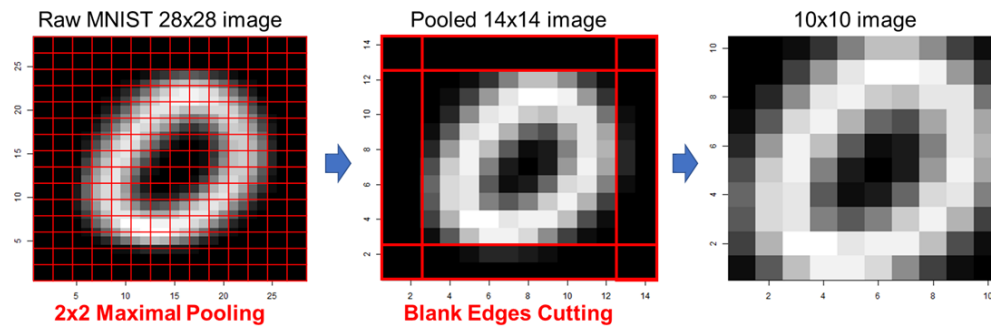


Figure 2: Data pre-processing for the MNIST dataset. First, the left subfigure shows the max-pooling step for reducing the image size. Next, the center subfigure shows the edge-cutting step for removing the noninformative pixels. Finally, the right subfigure shows the data pre-process result.

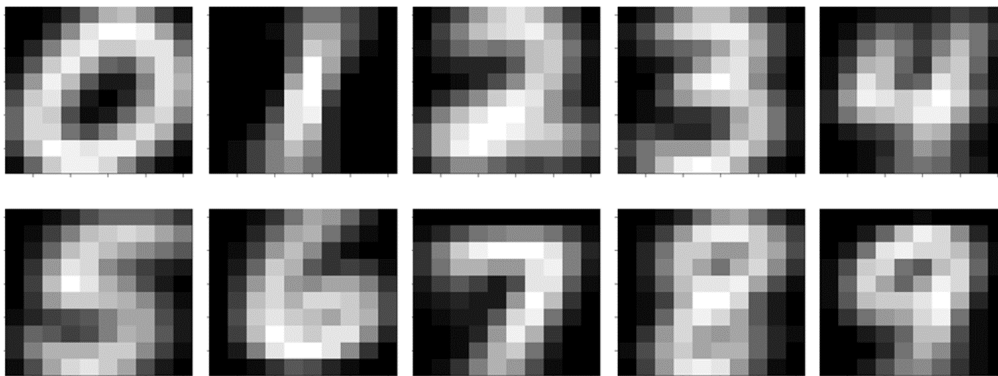


Figure 3: The mean plots of pre-processed images in the training dataset. The value at each pixel of the mean plot is the average grayscale value over the pre-processed images in the training dataset.

```
library(TensorTest2D)
data(mnist_mp2c2)
mnist_train <- mnist_mp2c2$train
mnist_test <- mnist_mp2c2$test
```

The aim of this data analysis is to recognize the digit for a given handwritten image using logistic regression. Here, we choose images of '2' and '5' for demonstration. Let $Y_i = 1$ if the i th image represents the digit '5', and $Y_i = 0$ if the i th image represents a '2'. The predictor matrix X_i here is a 10×10 matrix with its entries the pixel values of the handwritten image in grayscale. In the following, we first describe the data processing steps and provide the code being used to obtain our training data in the analysis. Our training data, `train_X`, is a $P \times Q \times n = 10 \times 10 \times 2000$ array, which contains subsets of 1,000 images of the digit '2' and 1,000 images of the digit '5' randomly chosen from the MNIST training set `mnist_train`. In this MNIST example, some pixels in the corners and on the edges take on the value zero across all handwritten images, which yields singularity when the alternating maximum likelihood algorithm is applied to the training data. To solve this problem, we can simply drop only those zero-valued pixels. However, doing so breaks the matrix form and hence low-rank model is no longer valid. Alternatively, we add independent standard normal noise to the images in our training data set that results in no significant harm to the prediction power, because the signal-noise ratio is high and the training set sample size is sufficiently large. Hereafter, we call images with random error the contaminated images.

```
library(abind)
# Draw image data of labels 2 and 5
x0_all <- mnist_train$image[, , which(mnist_train$label == 2)]
x1_all <- mnist_train$image[, , which(mnist_train$label == 5)]
# Random sampling from MNIST training set for each label
nSampleEach <- 1000
n0 <- dim(x0_all)[3]; n1 <- dim(x1_all)[3]
set.seed(2021)
s0 <- sample(1:n0, nSampleEach, replace = FALSE)
s1 <- sample(1:n1, nSampleEach, replace = FALSE)
```

```

# Normalizing image values into [-0.5, 0.5]
x0 <- x0_all[, ,s0]/255 - 0.5
x1 <- x1_all[, ,s1]/255 - 0.5
# Combine training data
train_X <- abind(x0, x1, along = 3)
# Add negligible noise for the images
# (so no constant zero values in one pixel over all covariate matrices)
set.seed(2021) # Set seed for reproducibility
train_n <- array((rnorm(prod(dim(train_X))), 0, 0.1)), dim(train_X))
train_Xn <- train_X + train_n # Contaminated images
# Define Y = 0 for label 2, and Y = 1 for label 5
train_y <- c(rep(0, dim(x0)[3]), rep(1, dim(x1)[3]))

```

In the package **TensorTest2D**, the function `tensorReg2D()` is also used for fitting the logistic tensor regression model to data:

$$\log \frac{\Pr(Y_i = 1 | \mathbf{X}_i)}{\Pr(Y_i = 0 | \mathbf{X}_i)} = \beta + \mathbf{X}_i \circ \mathbf{B}$$

Thus, a prediction for the digit presented in image \mathbf{X}_i is

$$\hat{Y}_i = \arg \max_{k \in \{0,1\}} \Pr(Y_i = k | \mathbf{X}_i) = I\{\Pr(Y_i = 1 | \mathbf{X}_i) > 0.5\},$$

where $I\{E\} = 1$, if E is true, and $I\{E\} = 0$, otherwise. To analyze the sampled data set, first, we feed the response variable, `train_y`, and the contaminated images, `train_Xn`, as inputs for model training. There is no auxiliary information available to further stratify the y_i 's, we specify a constant vector $\mathbf{W} = \text{matrix}(1, \text{length}(\text{train}_y), 1)$ of length n , and if a rank $R = 4$ model is needed, we set $n_R = 4$. (In fact, the rank-4 model is the best model in this logistic tensor regression.)

```

# Train the logistic tensor regression model
lgMdl <- tensorReg2D(y = train_y, X = train_Xn,
                    W = matrix(1, length(train_y), 1),
                    n_R = 4, family = "binomial",
                    opt = 1, max_ite = 100, tol = 10^(-7) )
# Print model summary (not run)
#summary(lgMdl)
# Print the p-values of the estimates
cat("FDR-adjusted p-values of B_pq:\n")

#> FDR-adjusted p-values of B_pq:

round(matrix(p.adjust(as.vector(lgMdl$B_PV), method = "fdr"), 10, 10), 3)

#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,] 0.263 0.830 0.114 0.243 0.019 0.480 0.595 0.629 0.491 0.830
#> [2,] 0.558 0.552 0.098 0.491 0.263 0.948 0.137 0.816 0.491 0.953
#> [3,] 0.923 0.927 0.029 0.012 0.999 0.017 0.204 0.050 0.471 0.008
#> [4,] 0.648 0.541 0.004 0.019 0.029 0.204 0.004 0.655 0.541 0.156
#> [5,] 0.293 0.491 0.055 0.004 0.381 0.004 0.101 0.024 0.081 0.006
#> [6,] 0.110 0.491 0.954 0.491 0.648 0.948 0.954 0.865 0.491 0.825
#> [7,] 0.023 0.652 0.889 0.019 0.489 0.491 0.110 0.188 0.042 0.029
#> [8,] 0.137 0.706 0.830 0.491 0.244 0.145 0.491 0.491 0.889 0.889
#> [9,] 0.655 0.034 0.655 0.977 0.083 0.114 0.019 0.629 0.706 0.471
#> [10,] 0.137 0.055 0.491 0.764 0.491 0.602 0.019 0.471 0.454 0.025

```

For binary classification problems, we can apply the function `plot()` of our package **TensorTest2D** in two ways. Similar to that Figure 1, we can make a plot first for the values of t -statistics for the pixels by using the `plot()` function as shown below this paragraph. Here, we adjust the p -values according to the approach in [Benjamini and Hochberg \(1995\)](#) by setting the parameter `method = "fdr"`. The resulting plot is shown in Figure 4, and most of the effective pixels can be found on the left half side of the plot.

```

plot(x = lgMdl, method = "fdr", alpha = 0.05, type = "tval",
     showlabels = TRUE, plot.legend = TRUE)

```

To understand which areas of an image that contribute the most information to classify between labels 2 and 5, we also add a meaningful background image by specifying an argument to the parameter

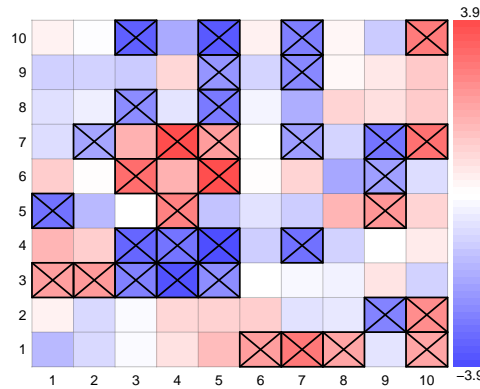


Figure 4: The image plot of the values for t-statistics of matrix covariate in the handwritten label data. The effective pixels identified by the logistic tensor regression model are marked out by the \otimes marks.

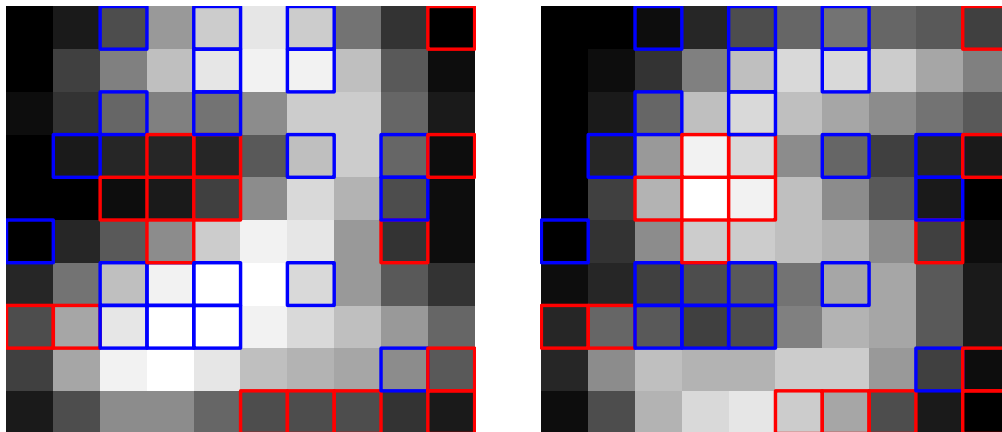


Figure 5: Effective pixels identified by the logistic tensor regression model. The important pixels to discriminate between labels 2 and 5 are marked by red and blue frames, which indicate positive and negative coefficients, respectively.

background for the function `plot()`. In this example, we show separately the mean image of label 2 and the mean image of label 5 as the background image by assigning the value `xm0` or `xm1` to the parameter `background`. To adjust the visual style of the background image, one can assign the value `gray(0, 1, 0.05)` to the parameter `col` to create a grayscale colour map for contrast. Please refer to `help("image")` for more detail on the options available when creating a plot. Our resulting plots are shown in Figure 5. On top of both images, there are marks for the important pixels with red and blue frames. A red rectangle indicates that the corresponding estimate in $\hat{\mathbf{B}}$ has a significant positive coefficient and a blue one highlights a significant negative coefficient. In our example, important pixels are found to locate majorly at the curvy parts of 2 and 5.

```
xm0 <- xm1 <- matrix(0, dim(train_X)[1], dim(train_X)[2])
# Background image: mean image of label 2
for (k in 1:dim(x0)[3]) {
  xm0 <- xm0 + (1/nSampleEach)*x0[, ,k]
}
# Background image: mean image of label 5
for (k in 1:dim(x1)[3]) {
  xm1 <- xm1 + (1/nSampleEach)*x1[, ,k]
}
# Draw for visualizing effective pixels for both background images
par(mfrow = c(1, 2), mar = c(1, 1, 1, 1))
plot(x = lgMdl, method = "fdr", alpha = 0.05, background = xm0,
      showlabels = FALSE, plot.legend = FALSE, col = gray(seq(0, 1, 0.05)))
plot(x = lgMdl, method = "fdr", alpha = 0.05, background = xm1,
      showlabels = FALSE, plot.legend = FALSE, col = gray(seq(0, 1, 0.05)))
```

We use the function `predict()` to predict the label for the new images in the testing data set. The

input data must be a 3-dimensional array of size $P \times Q \times n^*$, where n^* is the number of testing images. We note here that, one need to reshape the $P \times Q$ matrix object into the 3-dimensional array by the R command `array(x, c(P, Q, 1))`. The function `predict()` returns the predictions in two ways. By setting the option `type = "link"`, it returns the values of the linear predictors; and by setting `type = "response"`, it returns the expected values of response variable. For example, for our logistic regression model, the predictions are log-odds (odds ratios on logarithmic scale) if `type = "link"` is chosen, or they are the predicted probabilities of $Y = 1$ if `type = "response"` is chosen.

```
# Normalize image values of the testing data into [-0.5, 0.5]
tx0 <- mnist_test$image[,which(mnist_test$label == 2)]/255 - 0.5
tx1 <- mnist_test$image[,which(mnist_test$label == 5)]/255 - 0.5
# Combine testing data and assign the vector of the true responses
test_X <- abind(tx0, tx1, along = 3)
test_y <- c(rep(0, dim(tx0)[3]), rep(1, dim(tx1)[3]))
# Print some predictions with different settings of type
pred_link <- predict(lgMdl, test_X, type = "link")
pred_prob <- predict(lgMdl, test_X, type = "response")
head(round(pred_link, digits = 2))

#>      [,1]
#> [1,] -3.38
#> [2,] -16.24
#> [3,] -4.93
#> [4,] -5.38
#> [5,] -9.94
#> [6,] -6.41

head(round(pred_prob, digits = 4))

#>      [,1]
#> [1,] 0.0331
#> [2,] 0.0000
#> [3,] 0.0072
#> [4,] 0.0046
#> [5,] 0.0000
#> [6,] 0.0016

# Compute the prediction accuracy for the testing data
pred_test_y <- (pred_prob > .5)
cat(
  sprintf("Accuracy = %2.2f%",
          100*sum(pred_test_y == test_y)/length(test_y)))

#> Accuracy = 96.10%
```

In addition, we provide a visualization tool that works with other methods in variable selection for 2D images. For example, the penalized regression via lasso (Tibshirani, 1996) is one of the popular approaches. Below are the codes that we implemented to train a LASSO model, including the use of the function `cv.glmnet()` (Friedman et al., 2010). The object `l1B` is the 10×10 array of LASSO estimates. Since LASSO tends to shrink small coefficients to zero, we treat those image pixels with zero-valued coefficients to be irrelevant to distinguish between images of 2 and 5. To visualize the effective pixels identified by LASSO, our package `TensorTest2D` provides the function `draw.coef()` to produce the marked image similar to that in Figure 5. Different from the function `plot.tsglm()`, users need to provide an input as the markers for effective pixels. The markers in our example here are the LASSO estimates, and by specifying `marks = l1B`, the pixels with non-zero coefficients are then marked. In addition, by specifying `markstyle = "bi-dir"`, as shown in Figure 6, the pixels marked out with red rectangles correspond to those of positive LASSO estimates, and the pixels with blue rectangles are those of negative LASSO estimates.

```
library(glmnet)
# Vectorize the hand-written images
xv <- t(sapply(1:dim(train_X)[3], function(k) as.vector(train_X[, ,k])))
# Train the LASSO model using cross-validation
set.seed(2021) # Set seed for reproducibility
l1Mdl <- cv.glmnet(xv, train_y, family = "binomial", alpha = 1, standardize = FALSE)
# Draw the LASSO coefficients from the best model
```

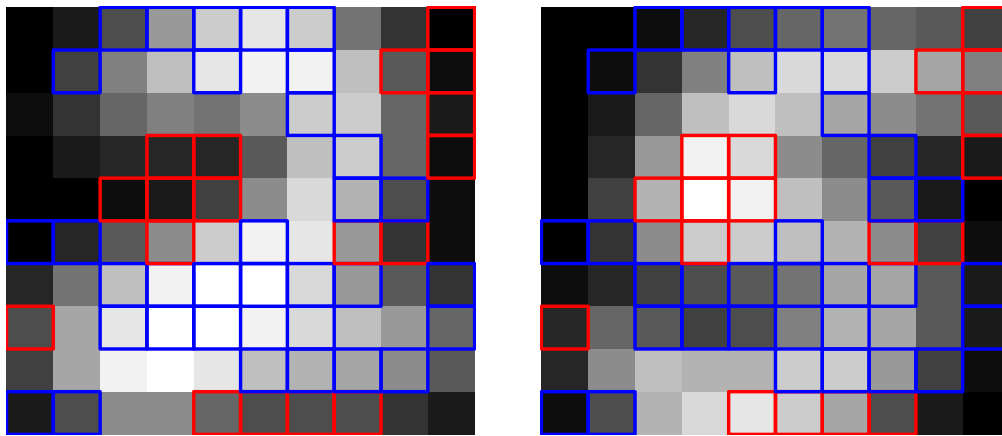


Figure 6: Effective pixels identified by the LASSO model. The important pixels to discriminate between labels 2 and 5 are marked by red and blue frames, which indicate positive and negative coefficients, respectively.

```

l1B <- matrix(l1Mdl$glmnet.fit$beta[,which.min(l1Mdl$cvm)], 10, 10)
# The LASSO estimates
print(round(l1B, digits = 3))

#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,] -0.247 0.000 0.253 0.000 -0.409 0.000 0.000 0.000 0.000 0.000
#> [2,] -0.559 0.000 0.000 0.000 -0.200 0.000 0.000 0.000 0.000 -0.691
#> [3,] 0.000 0.000 -0.248 -0.513 0.000 1.287 0.000 0.000 0.000 -0.914
#> [4,] 0.000 0.000 -1.526 -1.805 0.734 1.237 1.804 0.000 0.000 -0.254
#> [5,] 0.142 0.000 -0.593 -0.795 0.000 1.280 0.929 0.000 -0.699 -0.738
#> [6,] 1.625 -0.251 0.000 -1.429 -0.527 0.000 0.000 0.000 0.000 -0.592
#> [7,] 0.592 -0.136 -0.668 -0.554 0.000 0.000 0.000 -0.406 -0.397 -0.285
#> [8,] 0.074 -0.204 0.000 -0.757 0.265 -0.211 -1.082 0.000 0.000 0.000
#> [9,] 0.000 -0.574 0.000 0.000 0.360 -1.558 0.000 0.000 0.638 0.000
#> [10,] 0.000 0.000 -0.628 -0.144 0.000 0.000 0.479 1.340 0.303 0.845

# Draw for visualizing effective pixels identified by LASSO for both background images
par(mfrow = c(1, 2), mar = c(1, 1, 1, 1))
draw.coef(img = xm0, marks = l1B, markstyle = "bi-dir", showlabels = FALSE,
          plot.legend = FALSE, grids = FALSE, col = gray(seq(0, 1, 0.05)))
draw.coef(img = xm1, marks = l1B, markstyle = "bi-dir", showlabels = FALSE,
          plot.legend = FALSE, grids = FALSE, col = gray(seq(0, 1, 0.05)))

```

4 Summary

Issues in estimation and test of hypothesis that emerged from fitting regression models with predictor variables that has a matrix form are of our major interest. Low-rank modelling can be applied to improve the efficiency of estimation. In this line, we developed the R package [TensorTest2D](#) to conduct tensor regression analysis within the framework of generalized linear models. In addition to model estimation and hypothesis testing, this package also includes a visualization tool that can be used to indicate the positions of effective or significant pixels when the tensor predictor is of image data type.

Bibliography

- J. Barretina, G. Caponigro, N. Stransky, K. Venkatesan, A. A. Margolin, S. Kim, C. J. Wilson, J. Lehár, G. V. Kryukov, D. Sonkin, A. Reddy, M. Liu, L. Murray, M. F. Berger, J. E. Monahan, P. Morais, J. Meltzer, A. Korejwa, J. Jané-Valbuena, F. A. Mapa, J. Thibault, E. Bric-Furlong, P. Raman, A. Shipway, I. H. Engels, J. Cheng, G. K. Yu, J. Yu, P. Aspesi, M. de Silva, K. Jagtap, M. D. Jones, L. Wang, C. Hattton, E. Paleoscandolo, S. Gupta, S. Mahan, C. Sougnez, R. C. Onofrio, T. Liefeld, L. MacConaill, W. Winckler, M. Reich, N. Li, J. P. Mesirov, S. B. Gabriel, G. Getz, K. Ardlie, V. Chan, V. E. Myer, B. L. Weber, J. Porter, M. Warmuth, P. Finan, J. L. Harris, M. Meyerson, T. R. Golub,

- M. P. Morrissey, W. R. Sellers, R. Schlegel, and L. A. Garraway. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483:603–607, 2012. URL <https://doi.org/10.1038/nature11003>. [p154]
- Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: series B (Methodological)*, 57(1):289–300, 1995. [p158]
- S.-M. Chang, M. Yang, W. Lu, Y.-J. Huang, Y. Huang, H. Hung, J. C. Miecznikowski, T.-P. Lu, and J.-Y. Tzeng. Gene-set integrative analysis of multi-omics data using tensor-based association test. *Bioinformatics*, 03 2021. URL <https://doi.org/10.1093/bioinformatics/btab125>. [p152, 153, 154]
- M. Chen, S.-M. Chang, W. Lu, J.-Y. Tzeng, and P.-Y. Chen. *TensorTest2D: Fitting Second-Order Tensor Data*, 2021. URL <https://CRAN.R-project.org/package=TensorTest2D>. R package version 1.0.3. [p152]
- J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL <http://www.jstatsoft.org/v33/i01/>. [p160]
- H. Hung and C.-C. Wang. Matrix variate logistic regression model with application to EEG data. *Biostatistics*, 1(14):189–202, 07 2012. URL <https://doi.org/10.1093/biostatistics/kxs023>. [p153]
- R. A. Irizarry and A. Gill. *dslabs: Data Science Labs*, 2019. URL <https://CRAN.R-project.org/package=dslabs>. R package version 0.7.3. [p156]
- E. F. Lock. Tensor-on-tensor regression. *Journal of Computational and Graphical Statistics*, (27):638–647, 2018. doi: 10.1080/10618600.2017. [p152]
- E. F. Lock. *MultiwayRegression: Perform Tensor-on-Tensor Regression*, 2019. URL <https://CRAN.R-project.org/package=MultiwayRegression>. R package version 1.2. [p152]
- R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: series B (Methodological)*, 58(1):267–288, 1996. [p160]
- A. Wald. Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical Society*, 11 1943. URL <https://doi.org/10.2307/1990256>. [p154]
- W. Wang, J. Zeng, and X. Zhang. *TRES: Tensor Regression with Envelope Structure and Three Generic Envelope Estimation Approaches*, 2020. URL <https://CRAN.R-project.org/package=TRES>. R package version 1.1.3. [p152]
- H. Zhou, L. Li, and H. Zhu. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552, 2013. doi: 10.1080/01621459.2013.776499. URL <https://doi.org/10.1080/01621459.2013.776499>. [p152, 153]

Ping-Yang Chen
Chimes AI
12F, No. 201-8, Dunhua N. Rd., Songshan Dist.,
Taipei City 105076, Taiwan
pychen@chimes.ai

Hsing-Ming Chang
Department of Statistics and Institute of Data Science, National Cheng Kung University
1 University Road,
Tainan 70101, Taiwan
nckuhmchang@ncku.edu.tw

Yu-Ting Chen
Department of Statistics, Purdue University
250 N. University St, West Lafayette,
IN 47907, United States of America
150115011@gmail.com

Jung-Ying Tzeng
Department of Statistics and Bioinformatics Research Center, North Carolina State University

North Carolina State University
Raleigh NC, 27695, United States of America
jytzeng@ncsu.edu

Sheng-Mao Chang
Department of Statistics, National Taipei University
No. 151, University Rd., Sanxia Dist.,
New Taipei City 237303, Taiwan
Corresponding Author
smchang110@gm.ntpu.edu.tw

wavScalogram: An R Package with Wavelet Scalogram Tools for Time Series Analysis

by Vicente J. Bolós and Rafael Benítez,

Abstract In this work we present the **wavScalogram** R package, which contains methods based on wavelet scalograms for time series analysis. These methods are related to two main wavelet tools: the windowed scalogram difference and the scale index. The windowed scalogram difference compares two time series, identifying if their scalograms follow similar patterns at different scales and times, and it is thus a useful complement to other comparison tools such as the squared wavelet coherence. On the other hand, the scale index provides a numerical estimation of the degree of non-periodicity of a time series and it is widely used in many scientific areas.

1 Introduction

Since the works of Mallat (2008) and Daubechies (1992), wavelet analysis has become, in the last few decades, a standard tool in the field of time series analysis. Its ability to simultaneously analyze a signal in frequency space (scales) and in time, allows it to overcome many of the limitations that Fourier analysis presents for non-stationary time series. Furthermore, the algorithms for calculating the different wavelet transforms are characterized by their speed and ease of implementation.

There are currently many software packages that implement functions for wavelet analysis of time series (MATLAB's Wavelet Toolbox, Wavelab, etc.), and in recent years, the exponential growth of the R ecosystem has not been outside the field of wavelet analysis. Within CRAN there are many packages related to wavelet analysis for time series. Specifically, as collected in the *TimeSeries* Task View, the **wavelets** package (Aldrich, 2020), the **WaveletComp** and **biwavelet** packages (Roesch and Schmidbauer, 2018; Gouhier et al., 2021), the **mvLSW** package (Taylor et al., 2019) and other packages such as **hwwntest** (Savchev and Nason, 2018), **rwt** (Roebuck and Rice University's DSP group, 2022), **waveslim** (Whitcher, 2020) and **wavethresh** (Nason, 2022).

In this work we will describe in depth the **wavScalogram** package (Bolós and Benítez, 2021) (also mentioned in the *TimeSeries* Task View). In this package, methods based on the wavelet scalogram are introduced as defined in Benítez et al. (2010); Bolós et al. (2017, 2020). These methods are basically related to two main wavelet tools: the *windowed scalogram difference* and the *scale index*. The first one, the windowed scalogram difference, was introduced in Bolós et al. (2017). It allows to compare two time series at different scales and times, determining if their scalograms follow similar patterns. In this sense, it is a complement to other wavelet tools for comparing time series such as the squared wavelet coherence and the phase difference, since there are certain differences in time series that these measurements are not capable of detecting while the windowed scalogram difference can. The second tool is the scale index introduced in Benítez et al. (2010). It focuses on the analysis of the non-periodicity of a signal, giving a numerical measure of its degree of non-periodicity, taking the value 0 if the signal is periodic and a value close to 1 if the signal is totally aperiodic (for example, a purely stochastic signal). The scale index has been used in many scientific areas, being the evaluation of the quality of pseudo-random number generators the area where it has been used the most. In addition, the scale index also has a "windowed" version, in which the windowed scalogram is used to calculate the scale index instead, allowing to measure the evolution of the scale index over time, which is useful in the case of non-stationary time series (see Bolós et al. (2020)).

The article is organized as follows: In the next section, we describe the basics of the wavelet analysis and how to use them in the **wavScalogram** package. Then, a description of the wavelet scalogram and its implementation is given. The following sections are devoted to the windowed scalogram difference and the scale index, in its original and windowed versions. Finally we illustrate the use of the package with some examples in applied problems, such as the analysis of time series of sunspots or the use of the windowed scalogram difference in the clustering of time series, particularly the interest rate series of sovereign bonds.

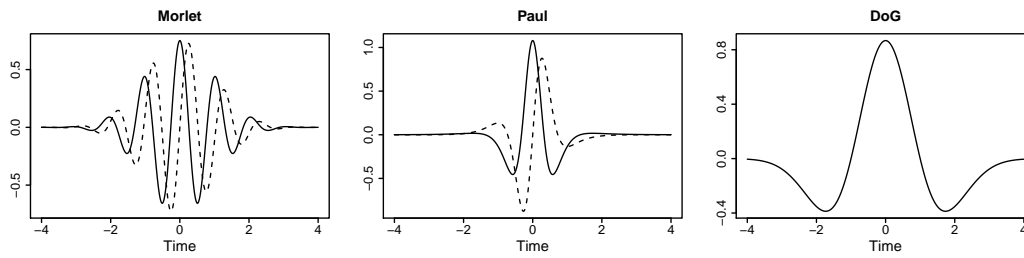


Figure 1: Real part (solid) and imaginary part (dashed) of Morlet, Paul and DoG wavelets for default parameter values, $\omega_0 = 6$ and $m = 4, 2$ respectively. Along with Haar, they are the most used in wavelet analysis.

2 Wavelet introduction

A wavelet (or mother wavelet) is a function $\psi \in L^2(\mathbb{R})$ with zero average (i.e. $\int_{\mathbb{R}} \psi = 0$), unit energy ($\|\psi\| = 1$, i.e. normalized) and centered in the neighborhood of $t = 0$ (Mallat, 2008). There exists a wide variety of wavelets but in this package we use the following, described in Torrence and Compo (1998) (see Figure 1):

- Morlet:

$$\psi_{\text{Morlet}}(t) = \pi^{-1/4} e^{i\omega_0 t} e^{-t^2/2}.$$

It is a plane wave modulated by a Gaussian, where the positive parameter ω_0 denotes the central dimensionless frequency. According to Farge (1992), the wavelet function must fulfil an admissibility condition, which for the Morlet wavelet is only accomplished if some correction factors are added. We take as default value $\omega_0 = 6$, for which those correction factors are negligible. Nevertheless, other choices of this parameter can be considered.

- Paul:

$$\psi_{\text{Paul}}(t) = \frac{(2i)^m m!}{\sqrt{\pi} (2m)!} (1 - it)^{-(m+1)},$$

where m is a positive integer parameter representing the order. By default, $m = 4$.

- Derivative of a Gaussian (DoG):

$$\psi_{\text{DoG}}(t) = \frac{(-1)^{m+1}}{\sqrt{\Gamma(m + \frac{1}{2})}} \frac{d^m}{dt^m} (e^{-t^2/2}),$$

where m is a positive integer parameter representing the derivative. By default, $m = 2$, that coincides with the Marr or Mexican hat wavelet.

Moreover, we have added:

- Haar, centered at 0:

$$\psi_{\text{Haar}}(t) = \begin{cases} 1 & \text{if } -\frac{1}{2} \leq t < 0, \\ -1 & \text{if } 0 \leq t < \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

This is the simplest wavelet, but it is not continuous.

Scaling a wavelet ψ by $s > 0$ and translating it by u , we create a family of unit energy “time-frequency atoms”, called *daughter wavelets*, $\psi_{u,s}$, as follows

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-u}{s}\right). \tag{1}$$

Remark 2.2.1 (Fourier factor). Usually, the Fourier wavelength of a daughter wavelet does not coincide with its scale s . Nevertheless, they are proportional, and this proportionality factor for converting scales into Fourier periods is called *Fourier factor*. This Fourier factor is taken $4\pi / (\omega_0 + \sqrt{2 + \omega_0^2})$, $4\pi / (2m + 1)$ and $2\pi / \sqrt{m + 1/2}$ for Morlet, Paul and DoG wavelets respectively (Torrence and Compo, 1998). For the default parameter values, the Fourier factor is approximately 1.033, 1.3963, and 3.9738 respectively. For Haar wavelet, the Fourier factor is 1.

Given a function $f \in L^2(\mathbb{R})$, that we will identify with a *signal* or *time series*, the *continuous wavelet transform* (CWT) of f at time u and scale $s > 0$ is defined as

$$\mathcal{W}f(u, s) = \int_{-\infty}^{+\infty} f(t) \psi_{u,s}^*(t) dt, \quad (2)$$

where $*$ denotes the complex conjugate. The CWT allows us to obtain the frequency components (or *details*) of f corresponding to scale s and time location u .

In practical situations, however, it is common to deal with finite signals. That is, given a time signal x , and a finite time interval $[0, T]$, we shall consider the finite sequence $x_n = x(t_n)$, for $n = 0, \dots, N$. Here, t_0, \dots, t_N is a discretization of the interval $[0, T]$, i.e. $t_n = nh$, being $h = T/N$ the time step. According to (1) and (2), the CWT of x at scale $s > 0$ is defined as the sequence

$$\mathcal{W}x_n(s) = h \sum_{i=0}^N x_i \psi_{n,s}^*(t_i), \quad (3)$$

where $n = 0, \dots, N$ and

$$\psi_{n,s}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t - t_n}{s}\right). \quad (4)$$

Note that $\psi_{n,s}(t)$ is in fact $\psi_{t_n,s}(t)$, but this abuse of notation between (1) and (4) is assumed for the sake of readability. Using Fourier transform tools, one can calculate (3) for all $n = 0, \dots, N$ simultaneously and efficiently (Torrence and Compo, 1998).

Remark 2.2.2 (Energy density). It is known that the CWT coefficients are biased in favour of large scales (Liu et al., 2007). Nevertheless, if the mother and daughter wavelets are normalized by the L^1 -norm (as in the **Rwave** package, by Carmona and Torresani (2021)) instead of the L^2 -norm (as in our package), this bias is not produced. Hence, to rectify the bias, the CWT in (2) and (3) can be multiplied by the factor $\frac{1}{\sqrt{s}}$. This rectification will be specially useful in some wavelet tools of our package that quantify the “energy density” of a signal, such as the wavelet power spectrum, the scalograms and the windowed scalogram difference. On the other hand, in the case of the scale index, this correction will not be advisable (see Remark 2.5.1). Usually, the wavelet tools of our package have a logical parameter called `energy_density` that switches this correction.

For computing the CWT of a time series x at a given set of scales, we use `cwt_wst`. For example,

```
# install.packages("wavScalogram")
library(wavScalogram)
h <- 0.1
N <- 1000
time <- seq(from = 0, to = N * h, by = h)
signal <- sin(pi * time)
scales <- seq(from = 0.5, to = 4, by = 0.05)
cwt <- cwt_wst(signal = signal, dt = h,
               scales = scales, powerscales = FALSE,
               wname = "DOG", wparam = 6)
```

computes the CWT of `signal` at scales from $s_a = 0.5$ to $s_b = 4$ using DoG wavelet with $m = 6$. The parameter `wname` indicates the wavelet used, and it can be “MORLET” (default value), “PAUL”, “DOG”, “HAAR” or “HAAR2”. The difference between these two last values is that “HAAR2” provides a more accurate but slower algorithm than the one provided by “HAAR”. Moreover, we can specify by means of `wparam` the value of the parameters ω_0 or m . As it has been stated before, the default values of these parameters are $\omega_0 = 6$ for Morlet wavelet, $m = 4$ for Paul wavelet and $m = 2$ for DoG wavelet.

If the set of scales is a base 2 power scales set (Torrence and Compo, 1998), the parameter `scales` can be a vector of three elements with the lowest scale s_a , the highest scale s_b , and the number of suboctaves per octave. This vector is internally passed to function `pow2scales` that returns the constructed base 2 power scales set. For example,

```
scales <- c(0.5, 4, 16)
cwt <- cwt_wst(signal = signal, dt = h, scales = scales, powerscales = TRUE)
```

computes the CWT of `signal` at scales from $s_a = 0.5$ to $s_b = 4$, with 16 suboctaves per octave. Since parameter `powerscales` is TRUE by default, it is not necessary to specify it in the function call. If `scales = NULL` (default value), then the function constructs the scales set automatically: s_a is chosen so that its equivalent Fourier period is $2h$ (Torrence and Compo, 1998), and $s_b = Nh/2r_w$, where r_w is the corresponding *wavelet radius*. Note that in this case s_b maximizes the length of the scales interval

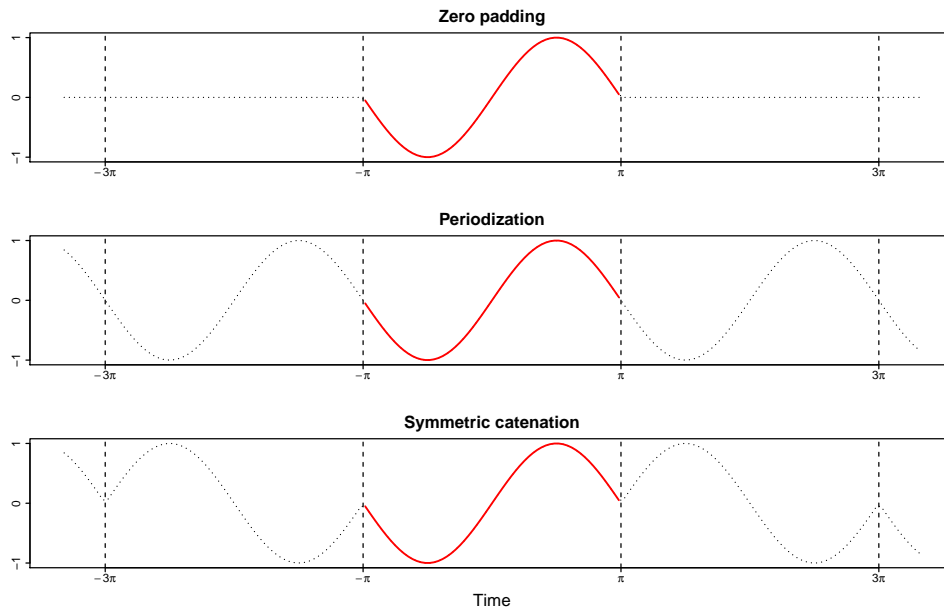


Figure 2: Different constructions of an infinite signal from a finite length signal $\sin(t)$ with $t \in [-\pi, \pi]$ (in red): padding time series with zeroes, using a periodization of the original time series, and using a symmetric catenation of the original time series. They determine the border effects.

taking into account the *cone of influence*. The wavelet radius and the cone of influence are defined and discussed in Remark 2.2.4.

The output `cwt` is a list containing the following fields:

- `coefs` is an $(N + 1) \times \text{length}(\text{scales})$ array (either real or complex depending on the wavelet used) containing the corresponding CWT coefficients, i.e. `cwt$coefs[i, j]` is the CWT coefficient at the i -th time and j -th scale.
- `scales` is the vector of scales used, either provided by the user or constructed by the function itself.
- `fourier_factor` is the scalar used to transform scales into Fourier periods (see Remark 2.2.1).
- `coi_maxscale` is a numeric vector of size $N + 1$ that defines the *cone of influence* (see Remark 2.2.4).

Remark 2.2.3 (Border effects). In (3) (or (2) for finite length signals) there appear *border effects* (or *edge effects*) when the support of the daughter wavelets is not entirely contained in the time domain $[t_0, t_N]$. In order to try to mitigate border effects, we can construct from the original time series x an infinite time series \bar{x} on $t_i = t_0 + ih$ for $i \in \mathbb{Z}$ and then we define

$$\bar{\mathcal{W}}x_n(s) = \mathcal{W}\bar{x}_n(s) = h \sum_{i \in \mathbb{Z}} \bar{x}_i \psi_{n,s}^*(t_i), \tag{5}$$

where $n = 0, \dots, N$. The most usual ways to construct \bar{x} are the following:

- Padding time series with zeroes: $\bar{x}_i = x_i$ if $i \in \{0, \dots, N\}$, and $\bar{x}_i = 0$ otherwise. In this case, (3) and (5) are equivalent, having $\bar{\mathcal{W}}x_n(s) = \mathcal{W}x_n(s)$.
- Using a periodization of the original time series: $\bar{x}_i = x_{i \bmod (N+1)}$.
- Using a symmetric catenation of the original time series: $\bar{x}_i = x_{i \bmod (N+1)}$ if $\lfloor \frac{i}{N+1} \rfloor$ is even, and $\bar{x}_i = x_{(N-i) \bmod (N+1)}$ if $\lfloor \frac{i}{N+1} \rfloor$ is odd.

Depending on the nature of x , it may be preferable to use one construction or another for minimizing the undesirable border effects (see Figure 2). For example, a periodization is advised for stationary short time series, and symmetric catenation for non-stationary short time series. On the other hand, for long time series, border effects are less important and then we can just pad with zeroes, i.e. use the original CWT given in (3).

How the border effects are treated by function `cwt_wst` is determined via the `border_effects` parameter. Possible values for this parameter are "BE" (raw border effects, padding with zeroes),

"PER" (periodization) and "SYM" (symmetric catenation), corresponding to the three options described above.

Remark 2.2.4 (Cone of influence). The *cone of influence* (CoI) is defined by the scales for which border effects become important at each time. The field `coi_maxscale` of the output in `cwt_wst` function contains, for each time, the maximum scale at which border effects are negligible, and consequently determines the CoI.

In order to compute the CoI, we have to set a criterion to distinguish between relevant and negligible border effects. In [Torrence and Compo \(1998\)](#), the CoI is defined by the ϵ -folding time for the autocorrelation of wavelet power spectrum (see the next section) at each scale s , and this ϵ -folding time is chosen so that the wavelet power spectrum for a discontinuity at the edge drops by a factor ϵ^{-2} . For Morlet and DoG wavelets, this ϵ -folding time is $\sqrt{2}s$, and for Paul wavelets is $s/\sqrt{2}$.

For wavelets with symmetric modulus such as Morlet, Paul and DoG, the ϵ -folding time at $s = 1$ is interpreted as a *wavelet radius* r_w that defines an *effective support* $[-r_w, r_w]$ for the mother wavelet. Therefore, the CoI is given by the scales from which the corresponding effective supports of the daughter wavelets $[u - sr_w, u + sr_w]$ are not entirely contained in the time domain.

The wavelet radius r_w determines the CoI in the different functions of this package by means of the parameter `waverad`. If it is NULL (default value) we consider $r_w = \sqrt{2}$ for Morlet and DoG wavelets, and $r_w = 1/\sqrt{2}$ for Paul wavelets, following [Torrence and Compo \(1998\)](#). On the other hand, we take $r_w = 0.5$ for Haar wavelet, i.e. we assume that its effective support is in fact its support. Nevertheless we can introduce a custom r_w for any wavelet, allowing us in this way to adjust the importance of border effects in the construction of the CoI. For example,

```
cwt <- cwt_wst(signal = signal, dt = h,
              wname = "DOG", wparam = 6, waverad = 2)
```

computes the CWT coefficients of `signal` for DoG wavelet with $m = 6$. Here, `cwt$coi_maxscale` is obtained assuming that the wavelet radius is $r_w = 2$. Note that the value of `waverad` does not affect the computation of the CWT coefficients.

3 Wavelet scalograms

The *wavelet power spectrum* of a signal $f \in L^2(\mathbb{R})$ at time u and scale $s > 0$ is defined as

$$\mathcal{WPS}f(u, s) = |\mathcal{W}f(u, s)|^2. \quad (6)$$

Analogously to (6), the wavelet power spectrum of a time series x at scale $s > 0$ is given by the sequence

$$\mathcal{WPS}x_n(s) = |\mathcal{W}x_n(s)|^2, \quad (7)$$

where $n = 0, \dots, N$.

We can plot the wavelet power spectrum of a time series x at a given set of scales through function `cwt_wst` if the parameter `makefigure` is TRUE (default value). There are other parameters regarding this plot:

- `time_values` is a vector that provides customized values in the time axis.
- `energy_density` is a logical parameter. If it is TRUE, it is plotted the wavelet power spectrum divided by the scales, according to Remark 2.2.2. By default, it is FALSE.
- `figureperiod` is a logical parameter that indicates if they are represented periods or scales in the y -axis (see Remark 2.2.1). By default, it is TRUE.
- `xlab`, `ylab`, `main`, `zlim` are parameters to customize the figure.

For example,

```
h <- 1 / 12
time1 <- seq(from = 1920, to = 1970 - h, by = h)
time2 <- seq(from = 1970, to = 2020, by = h)
signal <- c(sin(pi * time1), sin(pi * time2 / 2))
cwt_a <- cwt_wst(signal = signal, dt = h, time_values = c(time1, time2))
cwt_b <- cwt_wst(signal = signal, dt = h, time_values = c(time1, time2),
                 energy_density = TRUE)
```

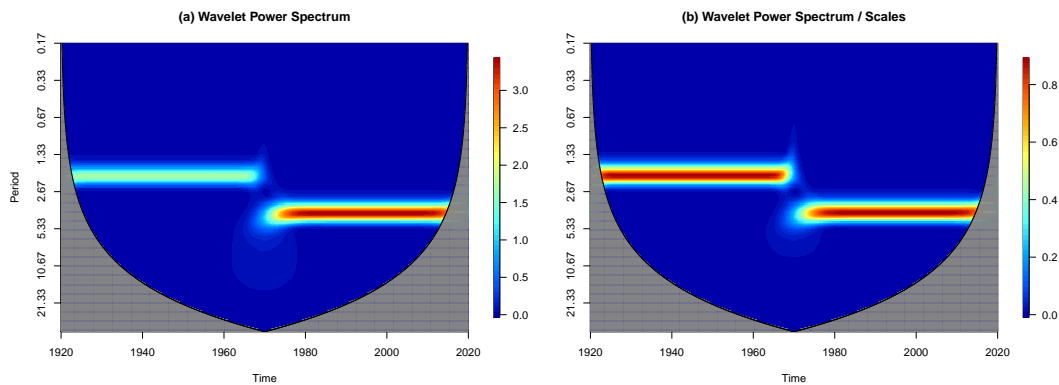



Figure 3: Wavelet power spectra of signal, non corrected (a) and corrected (b) via parameter `energy_density`. The Col is the shadowed region. This signal is the concatenation of two pure sinusoidal time series with the same amplitude and different periods. Note that even though both time series have the same amplitude, when the coefficients are not corrected, the magnitude of the wavelet power spectrum is biased in favour of large scales, while in the corrected version, this bias is not present.

plots Figure 3 (a) and (b) respectively. In this figure it is shown how the wavelet power spectrum is biased in favour of large scales, as it is pointed out in Remark 2.2.2. The parameter `energy_density` corrects it and so, values for different scales become comparable. Note that `energy_density` only affects the plot and hence, `cwt_a` is identical to `cwt_b`.

The *scalogram* of f at scale s is defined as

$$\mathcal{S}f(s) = \left(\int_{-\infty}^{+\infty} |\mathcal{W}f(u, s)|^2 du \right)^{1/2}. \quad (8)$$

It gives the contribution of each scale to the total “energy” of the signal and so, the notion of scalogram here is analogous to the spectrum of the Fourier transform. It is important to note that the term “scalogram” is often used to refer the wavelet power spectrum, but in this package, we call “scalogram” to (8).

If f is a finite length signal with time domain $I = [a, b]$, it is usual to consider a normalized version of the scalogram for comparison purposes, given by

$$\mathcal{S}f(s) = \left(\frac{1}{b-a} \int_a^b |\mathcal{W}f(u, s)|^2 du \right)^{1/2}. \quad (9)$$

Hence, according to (9), the (normalized) scalogram of x at scale s is given by

$$\mathcal{S}x(s) = \left(\frac{1}{N+1} \sum_{i=0}^N |\mathcal{W}x_i(s)|^2 \right)^{1/2}. \quad (10)$$

The normalization coefficient $1/(N+1)$ in (10) allows us to compare scalograms of time series with different lengths.

We can compute the (normalized) scalograms of a time series x at a given set of scales by means of function `scalogram`. This function follows the same rules as `cwt_wst` regarding the data entry, construction of scales, choice of the wavelet, border effects and application of Fourier factor. So, parameters `signal`, `dt`, `scales`, `powerscales`, `wname`, `wparam`, `waverad`, `border_effects`, `makefigure`, `figureperiod`, `xlab`, `ylab` and `main` are analogous to those in function `cwt_wst` with same default values. On the other hand, parameter `energy_density` is TRUE by default. For example,

```
sc_a <- scalogram(signal = signal, dt = h,
                 energy_density = FALSE)
```

computes the (normalized) scalogram of `signal` given by (10) at a base 2 power scales set constructed automatically and plots Figure 4 (a). The output `sc_a` is a list with the following fields:

- `scalog` is a vector of length `length(scales)` with the values of the (normalized) scalogram at each scale.
- `energy` is the total energy of `scalog` (i.e. its L^2 -norm) if parameter `energy_density` is TRUE.

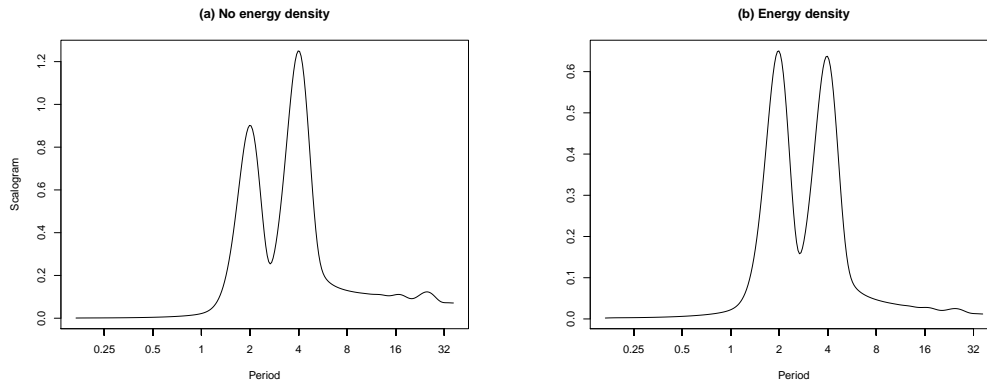


Figure 4: Original scalogram (a) and corrected scalogram representing an energy density measure (b), both relative to signal. As in Figure 3, it can be seen how the scalogram is biased in favour of large scales when the parameter `energy_density` is FALSE (plot (a)).

- `scales` and `fourier_factor` are analogous to those in the output of function `cwt_wst`.

If parameter `energy_density` is set to TRUE (default value), then the scalogram is divided by the square root of the scales, converting it into an energy density measure (see Remark 2.2.2). For example, if we write

```
sc_b <- scalogram(signal = signal, dt = h)
```

it plots Figure 4 (b), and then `sc_b$scalog` is in fact `sc_a$scalog / sqrt(scales)`.

Inner Scalogram

Given a compactly supported wavelet ψ and f a finite length signal with time domain $I = [a, b]$, the (normalized) inner scalogram of f at scale s is defined as

$$\mathcal{S}^{\text{inner}} f(s) = \left(\frac{1}{d(s) - c(s)} \int_{c(s)}^{d(s)} |\mathcal{W}f(u, s)|^2 du \right)^{1/2}, \quad (11)$$

where $[c(s), d(s)]$ is the maximal subinterval in I for which the support of $\psi_{u,s}$ is included in I for all $u \in [c(s), d(s)]$. Hence, according to (11), the (normalized) inner scalogram of x at scale s is given by

$$\mathcal{S}^{\text{inner}} x(s) = \left(\frac{1}{n_2(s) - n_1(s) + 1} \sum_{i=n_1(s)}^{n_2(s)} |\mathcal{W}x_i(s)|^2 \right)^{1/2},$$

where $\{n_1(s), \dots, n_2(s)\}$ is the maximal subset of time indices for which the support of $\psi_{i,s}$ is included in $[t_0, t_N]$ for all $i \in \{n_1(s), \dots, n_2(s)\}$.

This concept of inner scalogram can be extended to wavelets that do not have compact support, considering the effective support (see Remark 2.2.4) instead of the support. But we have to take into account that in this case, some theoretical results exposed in Benítez et al. (2010) may not hold.

We can compute the (normalized) inner scalograms of a time series x at a given set of scales by means of function `scalogram` setting the parameter `border_effects` equal to "INNER". Since Morlet, Paul and DoG wavelets are not compactly supported, it is considered the effective support given by the wavelet radius r_w .

Windowed Scalogram

The (normalized) windowed scalogram of f centered at time t with time radius $\tau > 0$ at scale s is defined as

$$\mathcal{WS}_\tau f(t, s) = \left(\frac{1}{2\tau} \int_{t-\tau}^{t+\tau} |\mathcal{W}f(u, s)|^2 du \right)^{1/2}. \quad (12)$$

It was introduced in Bolós et al. (2017) and it allows to determine the relative importance of the different scales around a given time point. According to (12), the (normalized) windowed scalogram

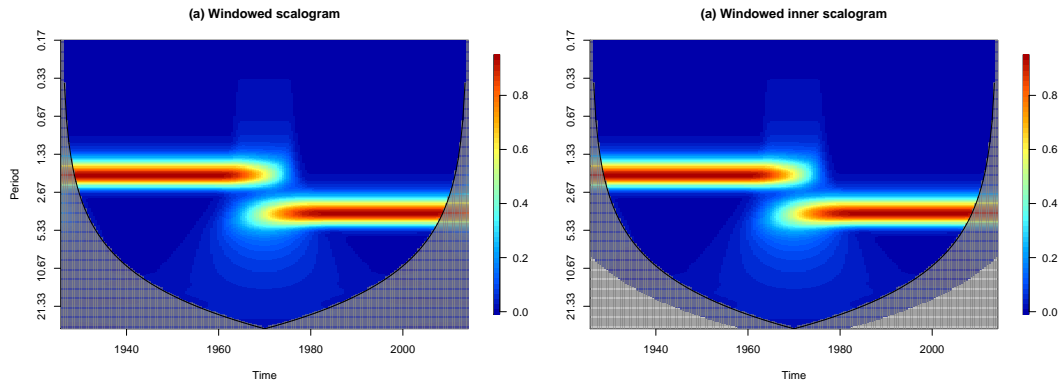


Figure 5: Windowed scalogram (a) and windowed inner scalogram (b) of signal. The Col is the shadowed region and, for the inner scalogram, the region where the scalogram cannot be computed is coloured in gray.

of x with time index radius $\tau \in \mathbb{N}$ at scale s is given by the sequence

$$\mathcal{WS}_\tau x_n(s) = \left(\frac{1}{2\tau + 1} \sum_{i=n-\tau}^{n+\tau} |\mathcal{W}x_i(s)|^2 \right)^{1/2}, \quad (13)$$

where $n = \tau, \dots, N - \tau$. In the particular case of $\tau = 0$, (13) coincides with $|\mathcal{W}x_n(s)|$.

We can compute the (normalized) windowed scalograms of a time series x at a given set of scales by means of function `windowed_scalogram`. Parameters `signal`, `dt`, `scales`, `powerscales`, `wname`, `wparam`, `waverad`, `border_effects`, `energy_density`, `makefigure`, `figureperiod`, `xlab`, `ylab` and `main` are analogous to those in function `scalogram`, and parameters `time_values` and `zlim` are analogous to those in function `cwt_wst`. For example,

```
wsc <- windowed_scalogram(signal = signal, dt = h,
                          windowrad = 72, delta_t = 6,
                          time_values = c(time1, time2))
```

computes the (normalized) windowed scalograms of `signal` with time index radius $\tau = \text{windowrad}$ at a base 2 power scales set constructed automatically. Moreover, it plots Figure 5 (a). If `windowrad` is `NULL` (default value), then it is set to $\lceil (N + 1)/20 \rceil$. The parameter `delta_t` is the index increment for the computation of the windowed scalograms, i.e. (13) is computed only for n from τ to $N - \tau$ by `delta_t`. If `delta_t` is `NULL` (default value) then it is taken $\lceil (N + 1)/256 \rceil$.

The output `wsc` is a list with the following fields:

- `tcentral` is the vector of times at which the windows are centered, i.e. the times of the form t_n where n goes from τ to $N - \tau$ by `delta_t`.
- `wsc` is a matrix of size $\text{length}(\text{tcentral}) \times \text{length}(\text{scales})$ containing the values of the windowed scalograms at each scale and at each central time.
- `windowrad` is the time index radius τ used.
- `scales`, `fourier_factor` and `coi_maxscale` are analogous to those in the output of function `cwt_wst`.

Windowed Inner Scalogram

The (normalized) windowed inner scalogram of f centered at time t with time radius $\tau > 0$ at scale s is defined as

$$\mathcal{WS}_\tau^{\text{inner}} f(t, s) = \left(\frac{1}{d(t, s) - c(t, s)} \int_{c(t, s)}^{d(t, s)} |\mathcal{W}f(u, s)|^2 du \right)^{1/2}, \quad (14)$$

where $[c(t, s), d(t, s)]$ is the maximal subinterval in $[t - \tau, t + \tau]$ for which the effective support of $\psi_{u, s}$ is included in I for all $u \in [c(t, s), d(t, s)]$. Then, the (normalized) windowed inner scalogram of x with

time index radius $\tau \in \mathbb{N}$ at scale s is given by the sequence

$$\mathcal{WS}_{\tau}^{\text{inner}} x_n(s) = \left(\frac{1}{n_2(n,s) - n_1(n,s) + 1} \sum_{i=n_1(n,s)}^{n_2(n,s)} |\mathcal{W}x_i(s)|^2 \right)^{1/2}, \quad (15)$$

where $\{n_1(n,s), \dots, n_2(n,s)\}$ is the maximal subset of time indices in $\{n - \tau, \dots, n + \tau\}$ for which the effective support of $\psi_{i,s}$ is included in $[t_0, t_N]$ for all $i \in \{n_1(n,s), \dots, n_2(n,s)\}$.

If `border_effects` is set to "INNER" in function `windowed_scalogram`, then the (normalized) windowed inner scalograms are computed. For example,

```
wsc <- windowed_scalogram(signal = signal, dt = h,
                          windowrad = 72, delta_t = 6,
                          border_effects = "INNER",
                          time_values = c(time1, time2))
```

computes the (normalized) windowed inner scalogram of `signal` with time index radius $\tau = \text{windowrad}$, at a base 2 power scales set constructed automatically, and plots Figure 5 (b). Note that in this figure, the "CoI line" is not a real CoI line, because if we consider inner scalograms, border effects are negligible. This line represents, at each time t_n , the maximum scale s such that $n_1(n,s) = n - \tau$ and $n_2(n,s) = n + \tau$, and coincides with the CoI line of the (normalized) windowed scalogram.

4 Windowed scalogram difference

The *windowed scalogram difference* (WSD) is a wavelet tool, introduced in Bolós et al. (2017), whose main objective is to compare time series by means of their respective windowed scalograms.

In order to consider differences between scalograms, it is convenient to use base 2 power scales (Bolós et al., 2017) and hence, we must redefine them by making a change of variable. Thus, for example, from (10), the (normalized) scalogram of a time series x at *log-scale* k should be given by

$$\mathcal{S}x(k) = \left(\frac{1}{N+1} \sum_{i=0}^N |\mathcal{W}x_i(2^k)|^2 \right)^{1/2}, \quad (16)$$

where $k \in \mathbb{R}$ is the binary logarithm of the scale. From now on, k will denote log-scales of scales s in the sense that $2^k = s$.

The windowed scalogram difference of two signals $f, g \in L^2(\mathbb{R})$ centered at (t, k) with time radius $\tau > 0$ and log-scale radius $\lambda > 0$ is defined as

$$\mathcal{WSD}_{\tau,\lambda} f g(t, k) = \left(\int_{k-\lambda}^{k+\lambda} \left(\frac{\mathcal{WS}_{\tau} f(t, \kappa) - \mathcal{WS}_{\tau} g(t, \kappa)}{\mathcal{WS}_{\tau} f(t, \kappa)} \right)^2 d\kappa \right)^{1/2}. \quad (17)$$

The commutative version of (17) is given by

$$\frac{1}{2} \left(\int_{k-\lambda}^{k+\lambda} \left(\frac{\mathcal{WS}_{\tau} f(t, \kappa)^2 - \mathcal{WS}_{\tau} g(t, \kappa)^2}{\mathcal{WS}_{\tau} f(t, \kappa) \mathcal{WS}_{\tau} g(t, \kappa)} \right)^2 d\kappa \right)^{1/2}. \quad (18)$$

From (17), the *windowed scalogram difference* (WSD) of two time series x, y centered at log-scale k with time index radius $\tau \in \mathbb{N}$ and log-scale radius $\lambda > 0$ is given by the sequence

$$\mathcal{WSD}_{\tau,\lambda} x y_n(k) = \left(\int_{k-\lambda}^{k+\lambda} \left(\frac{\mathcal{WS}_{\tau} x_n(\kappa) - \mathcal{WS}_{\tau} y_n(\kappa)}{\mathcal{WS}_{\tau} x_n(\kappa)} \right)^2 d\kappa \right)^{1/2}, \quad (19)$$

where $n = \tau, \dots, N - \tau$. However, in practice, we work with a finite interval of log-scales that is discretized into k_0, \dots, k_M with constant step. Thus, we can adapt (19) to this situation so that it can be written as

$$\mathcal{WSD}_{\tau,\lambda} x y_n(k_m) = \left(\frac{2\lambda + 1}{m_2 - m_1 + 1} \sum_{i=m_1}^{m_2} \left(\frac{\mathcal{WS}_{\tau} x_n(k_i) - \mathcal{WS}_{\tau} y_n(k_i)}{\mathcal{WS}_{\tau} x_n(k_i)} \right)^2 \right)^{1/2}, \quad (20)$$

where $\lambda \in \mathbb{N}$ is the log-scale index radius, $m_1 = \max\{0, m - \lambda\}$ and $m_2 = \min\{M, m + \lambda\}$. The factor $\frac{2\lambda+1}{m_2-m_1+1}$ is added to counteract the "border effects" in the log-scale interval that appear when

$m - \lambda < 0$ or $m + \lambda > M$, because in these cases, the number of addends is less than $2\lambda + 1$. Moreover, according to (18), a commutative version of (20) can be also considered.

We can compute the WSD (20) (or its commutative version) of two time series x, y of the same length and time step by means of function `wsd`. Parameters `dt`, `windowrad`, `delta_t`, `wname`, `wparam`, `waverad`, `border_effects`, `energy_density`, `makefigure`, `time_values`, `figureperiod`, `xlab`, `ylab`, `main` and `zlim` are analogous to those in function `windowed_scalogram`. For example,

```
set.seed(12345) # For reproducibility
N <- 1500
time <- 0:N
signal1 <- rnorm(n = N + 1, mean = 0, sd = 0.2) + sin(time / 10)
signal2 <- rnorm(n = N + 1, mean = 0, sd = 0.2) + sin(time / 10)
signal2[500:1000] = signal2[500:1000] + sin((500:1000) / 2)
wsd <- wsd(signal1 = signal1, signal2 = signal2,
           windowrad = 75, rdist = 14)
```

computes the commutative WSD of `signal1` and `signal2` centered at a log-scales set $\{k_0, \dots, k_M\}$ constructed automatically, with time index radius $\tau = \text{windowrad}$ and log-scale index radius $\lambda = \text{rdist}$. If `windowrad` is NULL (default value) then it is set to $\lceil (N + 1) / 20 \rceil$, and if `rdist` is NULL (default value) then it is set to $\lceil (M + 1) / 20 \rceil$. The log-scales set can be defined by parameter `scaleparam`, that must be a vector of three elements with the minimum scale, the maximum scale and the number of suboctaves per octave. Moreover, logical parameter `commutative`, whose default value is TRUE, determines if it is computed the commutative version of the WSD.

Remark 2.4.1 (Normalization). The WSD compares the patterns of the windowed scalograms of two time series determining if they give similar weights (or energy) to the same scales. Another tool for comparing two time series is the squared wavelet coherence (Torrence and Compo, 1998; Torrence and Webster, 1999), that measures the local linear correlation between them. So, these tools focus on different aspects: while the squared wavelet coherence does not take into account the magnitudes in the signals, for the WSD they are crucial. In fact, the WSD has sense only when the two time series considered are expressed in the same unit of measure or they are dimensionless. Otherwise, it will be necessary to somehow normalize the signals, but depending on the normalization method, some artifices could appear. For example, we can normalize the signals so that their scalograms have the same energy and, in this way, we can compare the relative contributions of each scale to the total energy. Another option is to normalize the signals so that their scalograms attain the same maximum value. Finally, it could be also useful to normalize the signals so that their scalograms reach the same value at a given reference scale.

The normalization method can be chosen through parameter `normalize` in function `wsd`. It can be set to "NO" (default value), "ENERGY", "MAX" or "SCALE", according to each normalization method exposed in Remark 2.4.1. In this last option, the reference scale must be given by parameter `refscale`.

Remark 2.4.2 (Near zero scalogram values). Some problems can arise in the WSD when a scalogram is zero or close to zero for a given log-scale because we are computing relative differences and hence, the WSD can take extremely high values or produce numerical errors. If we consider absolute differences this would not happen but, on the other hand, it would not be appropriate for scalogram values not close to zero. A solution is to establish a threshold for the scalogram values above which a relative difference is computed, and below which a difference proportional to the absolute difference is computed (the proportionality factor would be determined by requiring continuity). This threshold can be interpreted as the relative amplitude of the noise in the scalograms.

Another solution is to substitute the original windowed scalograms $\mathcal{WS}_\tau x_n(s)$, $\mathcal{WS}_\tau y_n(s)$ by

$$C + \left(1 - \frac{C}{\max}\right) \mathcal{WS}_\tau x_n(s), \quad C + \left(1 - \frac{C}{\max}\right) \mathcal{WS}_\tau y_n(s), \quad (21)$$

where

$$\max = \max_{n,s} \{\mathcal{WS}_\tau x_n(s), \mathcal{WS}_\tau y_n(s)\},$$

and $C \geq 0$ is a relatively small value, called *compensation* (see Figure 6).

Parameters `wscnoise` and `compensation` of function `wsd` allow us to deal with the near zero scalogram problem mentioned in Remark 2.4.2. The first one is a value in $[0, 1]$ and establishes the threshold from which a relative difference is computed. As particular cases, if it is 0 then relative differences are always done, and if it is 1 then absolute differences are always done. The default value is set to 0.02. The second one determines the compensation C of (21), which is set to 0 by default.

In practical situations, signals will be usually affected by random noises. Therefore it is necessary to determine whether the results obtained with the WSD are statistically significant or not. In this

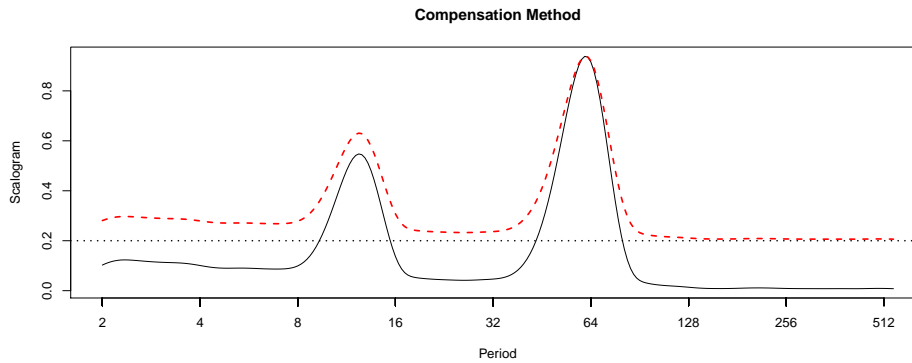


Figure 6: Illustration of the compensation method exposed in Remark 2.4.2 to deal with near zero scalogram values in the computation of the WSD. Original scalogram of signal2 (solid black line) is transformed into the compensated scalogram (dashed red line) for a compensation parameter $C = 0.2$.

package, we perform Monte Carlo simulations of the WSD (with the same parameter values) of random signals following a normal distribution with the same mean and standard deviation as the original ones. Then, we find the 95% and 5% quantiles to determine significantly high and low values respectively. The number of Monte Carlo simulations is set by parameter `mc_nrand` in function `wsd`, whose default value is 0 (no significant contours are computed). For example,

```
wsd <- wsd(signal1 = signal1, signal2 = signal2,
           mc_nrand = 100, parallel = TRUE)
```

computes the same WSD as before, but determines which values are significant using 100 Monte Carlo simulations, plotting Figure 7. Parameter `parallel` enables parallel computations improving considerably the execution time for high values of `mc_nrand`.

Finally, the output of `wsd` is a list with the following fields:

- `wsd` is a matrix of size $\text{length}(\text{tcentral}) \times \text{length}(\text{scales})$ containing the values of the WSD at each scale and at each central time.
- `rdist` is the log-scale index radius λ used.
- `signif95` and `signif05` are logical matrices of size $\text{length}(\text{tcentral}) \times \text{length}(\text{scales})$ that determine if the corresponding values of the `wsd` matrix are significantly high or low respectively, following the 95% and 5% quantiles method described above.
- `tcentral`, `scales`, `windowrad`, `fourier_factor` and `coi_maxscale` are analogous to those in the output of function `windowed_scalogram`.

With respect to the output image, it is plotted the base 2 logarithm of the inverse of the WSD because in this way high values represent small differences (i.e. high similarity) and low values represent large differences (i.e. low similarity) (Bolós et al., 2017).

5 Scale index and windowed scale index

Periodicity is one of the most basic characteristics to be determined in a time series study. Mathematically, the definition is clear: a time series f is periodic of period T whenever $f(t + T) = f(t)$ for all t , and a time series that fails to be periodic is a non-periodic signal. However, within this definition, there are very different types of non-periodic signals (e.g. stochastic, quasi-periodic, chaotic signals), and an interesting question to analyze is how much non-periodic a time series is. Within this regard, the *scale index* and the *windowed scale index* (Benítez et al., 2010; Bolós et al., 2020) are two wavelet tools that give a satisfactory answer to this question.

The *scale index* of a signal $f \in L^2(\mathbb{R})$ in the scale interval $[s_0, s_1]$ is defined as the quotient

$$i_{\text{scale}}f = \frac{\mathcal{S}f(s_{\min})}{\mathcal{S}f(s_{\max})}, \quad (22)$$

where $s_{\max} \in [s_0, s_1]$ is the smallest scale such that $\mathcal{S}f(s) \leq \mathcal{S}f(s_{\max})$ for all $s \in [s_0, s_1]$, and $s_{\min} \in [s_{\max}, 2s_1]$ is the smallest scale such that $\mathcal{S}f(s_{\min}) \leq \mathcal{S}f(s)$ for all $s \in [s_{\max}, 2s_1]$ (Benítez et al., 2010;

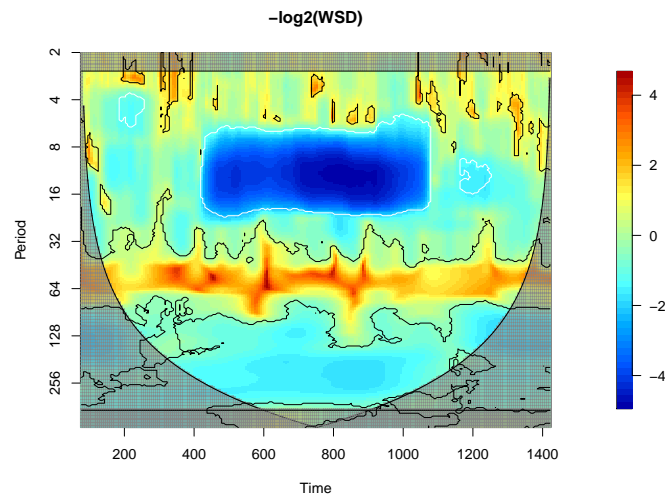


Figure 7: Base 2 logarithm of the inverse of the commutative WSD of `signal1` and `signal2` centered at a log-scales set constructed automatically, with time index radius $\tau = 75$ and log-scale index radius $\lambda = 14$. The significant contours are plotted in black (significantly high) and white (significantly low) lines, using 100 Monte Carlo simulations. Both time series are the same sinusoidal signal of period 20π plus different white noises with the same amplitude. Moreover, `signal2` has been manually modified for $500 \leq t \leq 1000$, with the addition of another pure sin signal of period 4π . The red band around period 20π corresponds to the period both signals have in common while the dark blue region around period 4π corresponds to the period both differ.

(Bolós et al., 2020). Hence, according to (22), the *scale index* of a time series x in the scale interval $[s_0, s_1]$ is given by

$$i_{\text{scale},x} = \frac{\mathcal{S}x(s_{\min})}{\mathcal{S}x(s_{\max})}, \quad (23)$$

where s_{\max} and s_{\min} are defined analogously.

The scale index is a quantity in $[0, 1]$ and measures the degree of non-periodicity of a signal in a given scale interval $[s_0, s_1]$: It is close to zero for periodic and quasi-periodic signals, and close to one for highly non-periodic signals. The choice of the scale interval $[s_0, s_1]$ is very important, and it should contain all the relevant scales that we want to study.

Remark 2.5.1 (No energy density). The correction exposed in Remark 2.2.2 should not be carried out because the scalogram of a white noise signal is more or less constant at all scales giving a scale index close to 1 for any scale interval $[s_0, s_1]$, and this is the property that we want to preserve. If, on the other hand, we apply the correction for converting the scalogram into an “energy density” measure, the scale index of a white noise signal would tend to zero as we increase s_1 and this is not desirable (Bolós et al., 2020).

We can compute the scale index of a time series x by means of function `scale_index`. Parameters `signal`, `dt`, `scales`, `powerscales`, `wname`, `wparam`, `waverad`, `border_effects`, `makefigure`, `figureperiod`, `xlab`, `ylab` and `main` are analogous to those in function `scalogram`. Note that, according to Remark 2.5.1, there is no parameter `energy_density` because scalograms must be computed without this correction. For example,

```
set.seed(12345) # For reproducibility
N <- 999
h <- 1 / 8
time <- seq(from = 0, to = N * h, by = h)
signal_si <- sin(pi * time) + rnorm(n = N + 1, mean = 0, sd = 2)
s0 <- 1
s1 <- 4
si <- scale_index(signal = signal_si, dt = h,
                  scales = c(s0, 2 * s1, 24), s1 = s1,
                  border_effects = "INNER", makefigure = FALSE)
```

computes the scale index of `signal_si` in the scale interval $[s_0, s_1]$ where $s_0 = 1$ and $s_1 = 4$. The parameter `scales` determines the scales set at which the scalograms are computed: In this case, it is

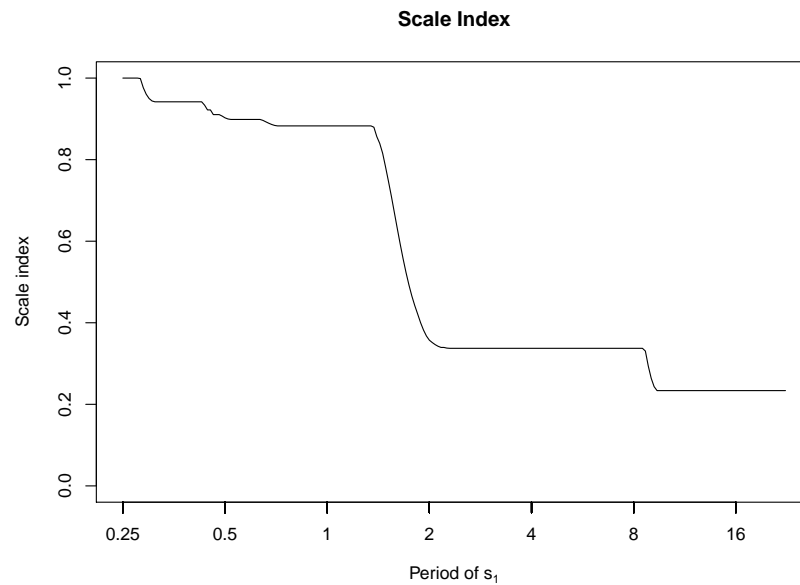


Figure 8: Scale indices of `signal_si` in scale intervals $[s_0, s_1]$ where s_0 is the scale whose equivalent Fourier period is 0.25 and s_1 varies. The signal is a pure sin of period 2 plus a noise term. Thus, for values of s_1 lower than 2, the scale index is very high because the scalogram still has not considered this period. At $s_1 = 2$, there is a sudden drop in the scale index and from that point onwards, as s_1 increases, the scale index decreases until it reaches a stable plateau (at 0.2 approximately) for large values of s_1 .

a base 2 power scales set from $s_a = s_0$ to $s_b = 2s_1$ with 24 suboctaves per octave. Note that we take $s_b = 2s_1$ according to the definition of the scale index, because s_b can not be lesser than $2s_1$ and there is no need for s_b to be greater than $2s_1$. Moreover, function `scale_index` takes s_0 equal to the lowest scale s_a always. If `scales = NULL` (default value), then the scalograms are computed at an automatically constructed set of scales with s_a equal to the scale whose equivalent Fourier period is $2h$, and $s_b = 2s_1$.

We can also compute the scale indices of a signal in scale intervals $[s_0, s_1]$ for different values of s_1 assigning a vector of scales to the parameter `s1`. Thus,

```
maxs1 <- 4
si <- scale_index(signal = signal_si, dt = h, scales = c(s0, 2 * maxs1, 24),
                 s1 = pow2scales(c(s0, maxs1, 24)),
                 border_effects = "INNER")
```

computes the scale indices of `signal_si` in scale intervals $[s_0, s_1]$ where $s_0 = 1$ and s_1 varies in a base 2 power scales set from 1 to 4 with 24 suboctaves per octave. Moreover, if `s1 = NULL` (default value), then `s1` is automatically computed as a base 2 power scales set from s_0 to $s_b/2$. If `scales` is also `NULL`, then $s_b = Nh/2r_w$ as usual, where r_w is the corresponding wavelet radius (see Remark 2.2.4). Hence,

```
si <- scale_index(signal = signal_si, dt = h, border_effects = "INNER")
```

computes the scale indices of `signal_si` in scale intervals $[s_0, s_1]$ where s_0 is the scale whose equivalent Fourier period is $2h$ and s_1 varies in a base 2 power scales set from s_0 to Nh/r_w . Moreover, it returns a plot like Figure 8. It is important to remark that if `s1` are not base 2 power scales then `powerscales` must be `FALSE`. For example,

```
si <- scale_index(signal = signal_si, dt = h,
                 s1 = seq(from = s0, to = maxs1, by = 0.1),
                 powerscales = FALSE, border_effects = "INNER")
```

computes the scale indices of `signal_si` in scale intervals $[s_0, s_1]$ where $s_0 = 1$ and s_1 varies linearly from 1 to 4 with step 0.1. In this case, since `scales` are not given, they are constructed automatically in a linear form, since `powerscales` must be `FALSE`.

Alternatively, we can compute the scale indices directly from a scalogram instead of giving the original signal by means of parameter `scalog`. In this case, we must give the scales at which the scalogram has been computed. Thus, we can compute the scale indices of an artificially constructed

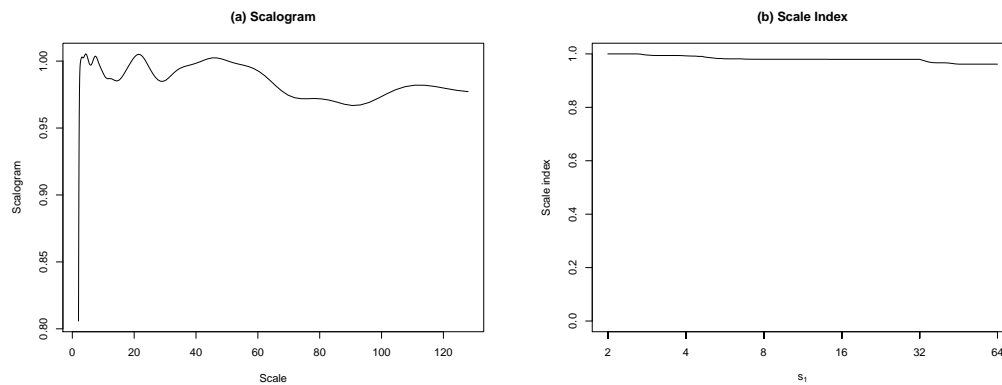


Figure 9: (a) Average of 100 scalograms of noise signals. (b) Scale indices computed from this averaged scalogram. The scale indices are very close to 1, indicating a high degree of non-periodicity.

scalogram which does not necessarily have to correspond to any signal. For example, we can compute the scale indices corresponding to the average of 100 scalograms of noise signals:

```
set.seed(12345) # For reproducibility
N <- 1000
nrand <- 100
X <- matrix(rnorm(N * nrand), nrow = N, ncol = nrand)
scales = pow2scales(c(2, 128, 24))
ns = length(scales)
sc_list <- apply(X, 2, scalogram, scales = scales, border_effects = "INNER",
                energy_density = FALSE, makefigure = FALSE)
sc_matrix <- matrix(unlist(lapply(sc_list, "[[", "scalogram")),
                  nrow = ns, ncol = nrand)
sc_mean <- apply(sc_matrix, 1, mean)
s1 = pow2scales(c(2, 64, 24))
si_mean <- scale_index(scalog = sc_mean, scales = scales, s1 = s1,
                      figureperiod = FALSE, plot_scalog = TRUE)
```

This code also returns figures like those in Figure 9. The logical parameter `plot_scalog` is used for plotting the scalogram from which the scale indices are computed.

The output of `scale_index` is a list with the following fields:

- `si` is a vector with the scale indices, for each value of s_1 .
- `s0` is the scale s_0 .
- `s1` is a vector with the scales s_1 .
- `smax` and `smin` are vectors with the scales s_{\max} and s_{\min} respectively, for each value of s_1 .
- `scalog` is the the scalogram $\mathcal{S}x$ from which the scale indices are computed.
- `scalog_smax` and `scalog_smin` are vectors with the scalogram values $\mathcal{S}x(s_{\max})$ and $\mathcal{S}x(s_{\min})$ respectively, for each value of s_1 .
- `fourierfactor` is the scalar used to transform scales into Fourier periods (see Remark 2.2.1).

Windowed Scale Index

As was mentioned in the introduction, wavelet analysis is a very useful tool for non-stationary time series. If we are interested in analyzing the non-periodicity of a non-stationary time series, we should be aware that the scale index is going to give us a single number between 0 and 1 which represents the degree of non-periodicity of the signal in the overall time interval of interest. However we may be interested in how this degree of non-periodicity is changing along this interval. To this aim, Bolós et al. (2020) introduced the *windowed scale index*, which uses the windowed scalogram in order to obtain scale indices for different time and scale intervals.

In particular, the *windowed scale index* of f in the scale interval $[s_0, s_1]$ centered at time t with time radius $\tau > 0$ is defined as

$$w_{i_{\text{scale}, \tau}} f(t) = \frac{\mathcal{W}\mathcal{S}_{\tau} f(t, s_{\min})}{\mathcal{W}\mathcal{S}_{\tau} f(t, s_{\max})}, \quad (24)$$

where, analogously to (22), s_{\max} is the smallest scale such that $\mathcal{WS}_{\tau}f(t, s) \leq \mathcal{WS}_{\tau}f(t, s_{\max})$ for all $s \in [s_0, s_1]$, and s_{\min} is the smallest scale such that $\mathcal{WS}_{\tau}f(t, s_{\min}) \leq \mathcal{WS}_{\tau}f(t, s)$ for all $s \in [s_{\max}, 2s_1]$ (Bolós et al., 2020). Finally, according to (24), the *windowed scale index* of a time series x in the scale interval $[s_0, s_1]$ with time index radius $\tau \in \mathbb{N}$ is given by the sequence

$$wi_{\text{scale}, \tau} x_n = \frac{\mathcal{WS}_{\tau} x_n(s_{\min})}{\mathcal{WS}_{\tau} x_n(s_{\max})},$$

where $n = \tau, \dots, N - \tau$, and s_{\max}, s_{\min} are defined analogously to (24).

Remark 2.5.2 (Inner scalograms). Although in the computation of the scale index it is recommended the use of (normalized) inner scalograms in order to fulfil some theoretical results and avoid border effects, this recommendation is less important in the case of the windowed scale index, because for long time series and relatively small time radii there would be no relevant border effects in most of the windowed scalograms.

By means of function `windowed_scale_index`, we can compute the windowed scale index of a time series x . As usual, parameters `signal`, `dt`, `scales`, `powerscales`, `windowrad`, `delta_t`, `wname`, `wparam`, `waverad`, `border_effects`, `makefigure`, `time_values`, `figureperiod`, `xlab`, `ylab`, `main` and `zlim` are analogous to those in function `windowed_scalogram`. Moreover, parameter `s1` is analogous to that in function `scale_index`. For example,

```
set.seed(12345) # For reproducibility
s0 <- 1
s1 <- 4
signal1_wsi <- sin(pi * time[1:500]) + rnorm(n = 500, mean = 0, sd = 2)
signal2_wsi <- sin(pi * time[501:1000] / 2) + rnorm(n = 500, mean = 0, sd = 0.5)
signal_wsi <- c(signal1_wsi, signal2_wsi)
wsi <- windowed_scale_index(signal = signal_wsi, dt = h,
                           scales = c(s0, 2 * s1, 24), s1 = s1,
                           windowrad = 50,
                           time_values = time)
```

computes the windowed scale index of `signal_wsi` in a scale interval $[s_0, s_1]$ where $s_0 = 1$ and $s_1 = 4$. The time index radius τ is given by the parameter `windowrad`. If it is `NULL` (default value), then it is set to $\lceil (N + 1)/20 \rceil$ that, in this case, coincides with the value of `windowrad`. Moreover, it returns a plot like Figure 10.

We can compute the windowed scale indices for different values of s_1 assigning a vector of scales to the parameter `s1`. It is important to remark that if `s1` are not base 2 power scales, then `powerscales` must be `FALSE`. If `s1 = NULL` and/or `scales = NULL` (default values), then they are automatically computed in the same way as it is done in function `scale_index`. So,

```
wsi <- windowed_scale_index(signal = signal_wsi, dt = h,
                           time_values = time)
```

computes the windowed scale indices of `signal_wsi` in scale intervals $[s_0, s_1]$ where s_0 is the scale whose equivalent Fourier period is $2h$ and s_1 varies in a base 2 power scales set from s_0 to Nh/r_w . The time index radius τ is taken automatically as $\lceil (N + 1)/20 \rceil = 50$. It also returns a plot, like Figure 11.

Alternatively, we can compute the windowed scale indices directly from a windowed scalogram instead of giving the original signal by means of parameter `wsc`. This parameter must be equal to a matrix of size (number of central times) \times (number of scales), as it is returned by the `windowed_scalogram` function. In this case, we must give the scales at which the windowed scalogram `wsc` has been computed and, in addition, we can give the cone of influence by means of parameter `wsc_coi`, that must be a vector containing the values of the maximum scale at each central time from which there are border effects in `wsc`. Thus, we can compute the windowed scale indices of an artificially constructed windowed scalogram as it was shown in the case of the scale index. Taking the same example, we can compute the windowed scale indices corresponding to the average of 100 windowed scalograms of noise signals:

```
set.seed(12345) # For reproducibility
N <- 1000
nrand <- 100
X <- matrix(rnorm(N * nrand), nrow = N, ncol = nrand)
scales = pow2scales(c(2, 128, 24))
ns = length(scales)
wsc_list <- apply(X, 2, windowed_scalogram, scales = scales,
                 energy_density = FALSE, makefigure = FALSE)
```

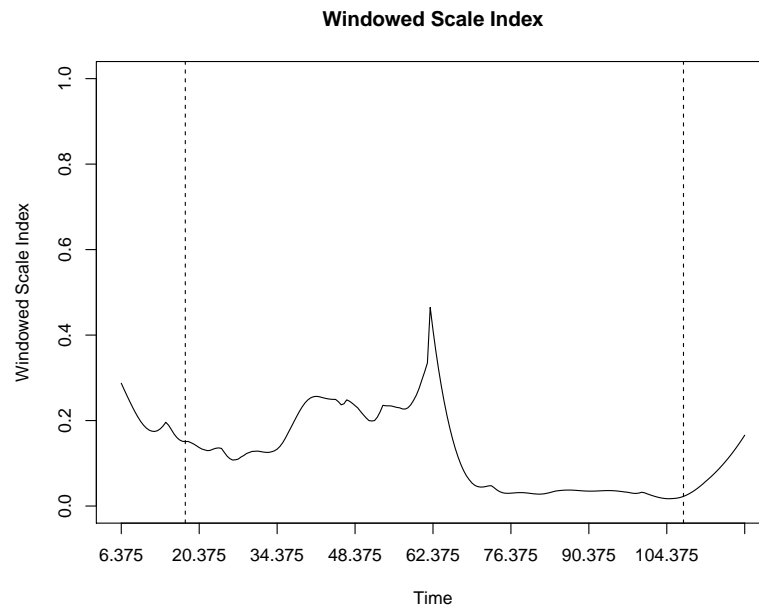


Figure 10: Windowed scale index of `signal_wsi` in a scale interval $[1, 4]$ and with time index radius $\tau = 50$. The dashed vertical lines represent the CoI limits. This time series is the concatenation of two sinusoidal signals of periods 2 and 4, modified with two white noises of different variance. In the first part, where the noise has a higher standard deviation, the windowed scale index is also higher. Moreover, it can be seen how the windowed scale index captures the moment of change in the noise.

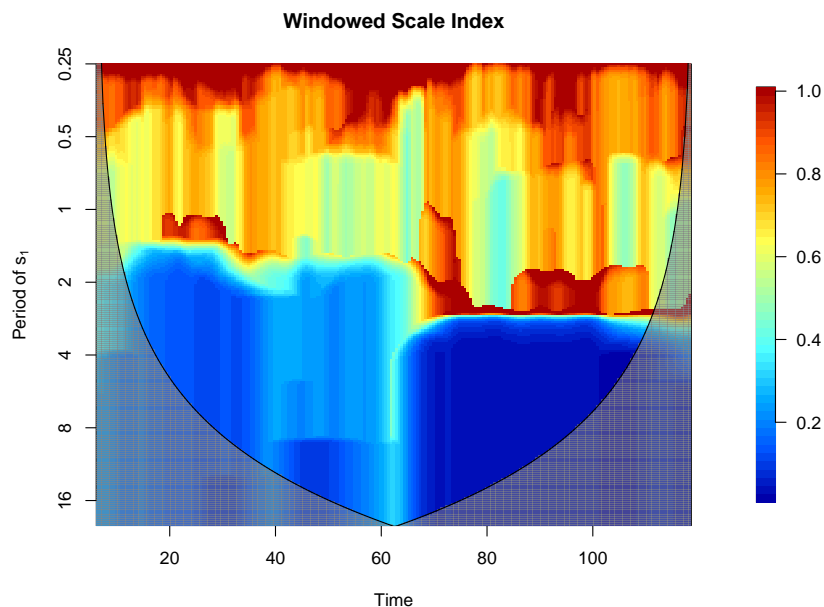


Figure 11: Windowed scale indices of `signal_wsi` in scale intervals $[s_0, s_1]$ where s_0 is the scale whose equivalent Fourier period is 0.25 and s_1 varies, with time index radius $\tau = 50$. This plot also shows that s_1 should be at least 4 for the scale indices to capture all relevant periods.

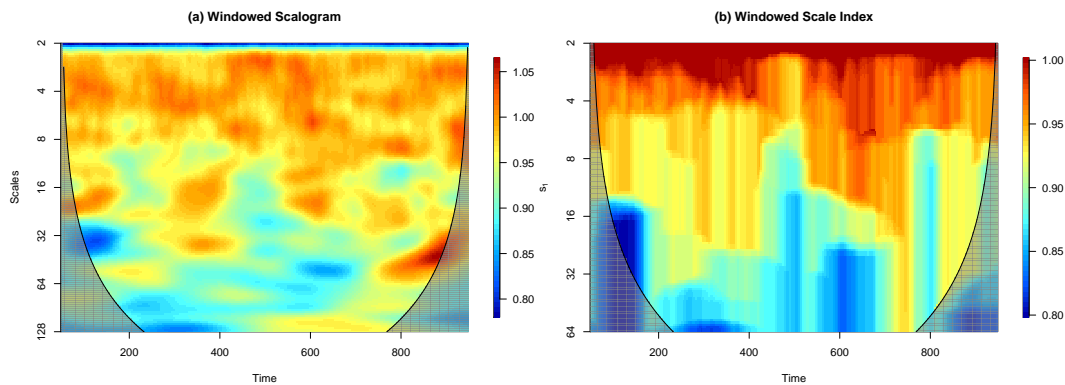


Figure 12: (a) Average of 100 windowed scalograms of noise signals. (b) Windowed scale indices computed from this averaged windowed scalogram. The windowed scale indices are always close to 1, indicating a high degree of non-periodicity.

```
tcentral <- wsc_list[[1]]$tcentral
ntc <- length(tcentral)
wsc_matrix <- array(unlist(lapply(wsc_list, "[", "wsc")), c(ntc, ns, nrand))
wsc_mean <- apply(wsc_matrix, 1:2, mean)
wsc_coi <- wsc_list[[1]]$coi_maxscale
wsi_mean <- windowed_scale_index(wsc = wsc_mean, wsc_coi = wsc_coi,
                                scales = scales, time_values = tcentral,
                                figureperiod = FALSE, plot_wsc = TRUE)
```

This code also returns figures like those in Figure 12. The logical parameter `plot_wsc` is used for plotting the windowed scalogram from which the windowed scale indices are computed.

The output of `windowed_scale_index` is a list with the following fields:

- `wsi` is a matrix of size $\text{length}(tcentral) \times \text{length}(s1)$ with the windowed scale indices at each s_1 and at each central time.
- `wsc` is a matrix of size $\text{length}(tcentral) \times \text{length}(scales)$ with the windowed scalograms from which the windowed scale indices are computed. Note that scales greater than $2 \cdot \max(s_1)$ are not necessary and they are internally removed from scales.
- `s0`, `s1`, `smax`, `smin`, `scalog_smax` and `scalog_smin` are analogous to those in the output of function `scale_index`.
- `tcentral`, `windowrad`, `fourierfactor` and `coi_maxscale` are analogous to those in the output of function `windowed_scalogram`.

6 Examples and applications

Windowed scalogram difference and clustering

As an application, we are going to show an example of how to define a dissimilarity measure from the windowed scalogram difference (WSD), which can then be applied to perform time series clustering. We are going to use the `interest.rates` time series from package `TSclust` (Montero and Vilar, 2014), which consists on 215 observations of the monthly long-term interest rates (10-year bonds) from January 1995 to November 2012 of several countries.

First, we define the `returns` time series for each country and then we compute the corresponding WSD of any pair of countries (see Figure 13). Next, we define the dissimilarity measure as the binary logarithm of the WSD mean plus 1 (in order to avoid negative distances). Finally, we plot the hierarchical clusters according to this dissimilarity measure (see Figure 14).

When defining the dissimilarity measure, we can restrict the WSD to only some areas instead of considering it entirely. For example, if we want to study the relationships between the different countries from the beginning of the century to the 2008 crisis at long-term scales, then we could only take into account the WSD area between 2001 and 2007, considering exclusively scales greater than 2 years. On the other hand, if border effects are relevant, only the WSD zone outside the cone of

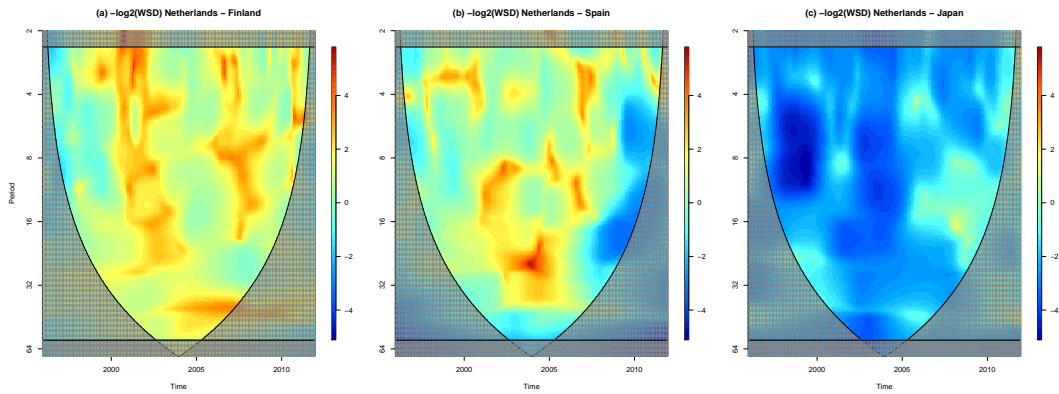


Figure 13: Plots of base 2 logarithms of the inverse of the commutative WSD of returns of Netherlands and (a) Finland, (b) Spain and (c) Japan. The corresponding dissimilarity measures of these pairs are 0.7395, 1.6279 and 2.819 respectively. Red zones indicate time-scale regions where the two signals are more similar, while blue zones correspond to less similarity between the signals. Note that Netherlands and Finland are two countries whose economies are similar in both time and scale (plot (a)) but, on the other hand, Netherlands has a very different economic behaviour than Japan (plot (c)). The big blue spot in plot (b) corresponds to the 2008 financial crisis, which hit Spain harder than the Netherlands.

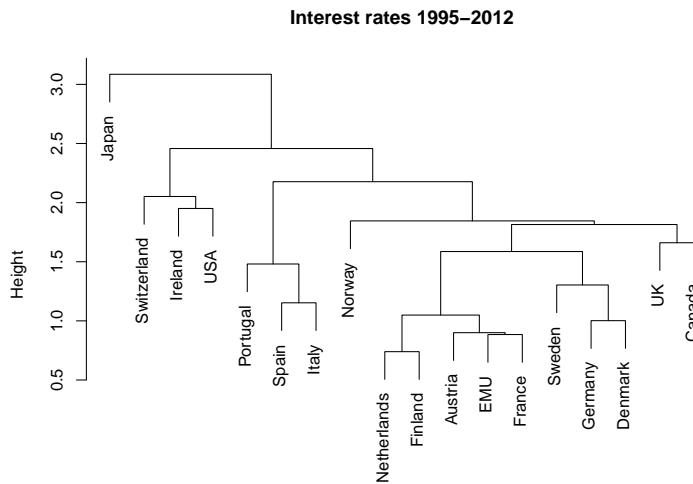


Figure 14: Hierarchical clustering of several countries according to their interest rates from 1995 to 2012. Similar countries are close together in the diagram.

influence could be considered. However, in our example, border effects do not substantially alter the clustering result.

```
library(wavScalogram)
library(TSclust)
data("interest.rates")
returns <- apply(interest.rates, MARGIN = 2, function(x) diff(log(x)))
Nsignals <- ncol(returns)
countries <- colnames(returns)
M <- Nsignals * (Nsignals - 1) / 2 # Number of pairings
auxpair <- vector(mode = "list", M)
k <- 1
for (i in 1:(Nsignals - 1)) {
  for (j in (i + 1):Nsignals) {
    auxpair[[k]] <- c(i, j)
    k <- k + 1
  }
}
fwsd <- function(x) wsd(signal1 = returns[, x[1]],
                       signal2 = returns[, x[2]],
                       makefigure = FALSE)
Allwsd <- lapply(auxpair, FUN = fwsd)
ntimes <- length(Allwsd[[1]]$tcentral)
nscales <- length(Allwsd[[1]]$scales)
area <- ntimes * nscales
meanwsd <- rep(0, M)
for (i in 1:M) {
  meanwsd[i] <- sum(Allwsd[[i]]$wsd) / area
}
d1 <- matrix(0, Nsignals, Nsignals)
d1[lower.tri(d1, diag = FALSE)] <- log2(meanwsd + 1)
dm1 <- as.dist(t(d1) + d1)
names(dm1) <- countries
plot(hclust(dm1), main = "Interest rates 1995-2012", xlab = "", sub = "")
```

Sunspots

In the next example we are going to illustrate the different tools of **wavScalogram** on the most famous *sunspot number time series* and how to use them in order to find the sunspots period, which is estimated to be around 11 years.

Let us consider the `sunspot.month` R dataset consisting on monthly numbers of sunspots from 1749 to present. Firstly, we can estimate the sunspots period by means of the scale at which the scalogram reaches its maximum. Using this criterion, we obtain that the sunspots period is 10.3254 approximately (see Figure 15 (b)). For this method, it is recommended that `energy_density = TRUE` since otherwise, larger scales would be over-estimated. Note that the wavelet power spectrum and the windowed scalograms present, as expected, horizontal bands of high values precisely around the scale 10.3254 (see Figure 15 (a) and (c)). Hence, they can be used to estimate a sunspot period that depends on time.

On the other hand, we can also estimate the sunspots period by means of the scale at which the scale index reaches its minimum. Contrary to the previous case and according to Remark 2.5.1, it is recommended that `energy_density = FALSE` for computing scale indices (see Figure 16). Using this criterion, we obtain that the sunspots period is 11.1215, approximately (see Figure 17 (a)). Therefore, the windowed scale index can also be used, analogously to the scalogram and the windowed scalogram, to estimate sunspots periods depending on time (see Figure 17 (b)).

Bibliography

- E. Aldrich. *wavelets: Functions for Computing Wavelet Filters, Wavelet Transforms and Multiresolution Analyses*, 2020. URL <https://CRAN.R-project.org/package=wavelets>. R package version 0.3-0.2. [p164]
- R. Benítez, V. J. Bolós, and M. E. Ramírez. A wavelet-based tool for studying non-periodicity. *Computers*

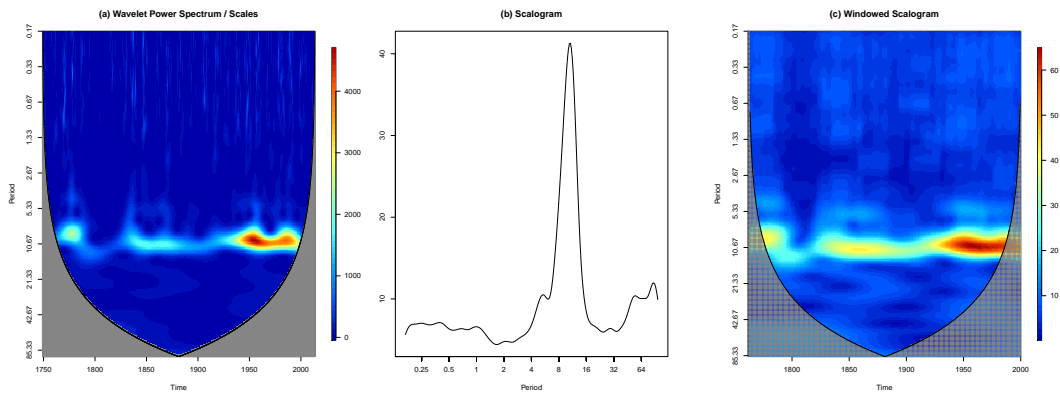


Figure 15: (a) Wavelet power spectrum divided by scales, (b) scalogram, and (c) windowed scalogram of the sunspots time series, with `energy_density = TRUE`. These plots show how the scalogram can be used for determining the sunspots period. In plots (a) and (c) the yellow-red band should be centered in the sunspots period, while in plot (b), this period should be given by the peak in the scalogram.

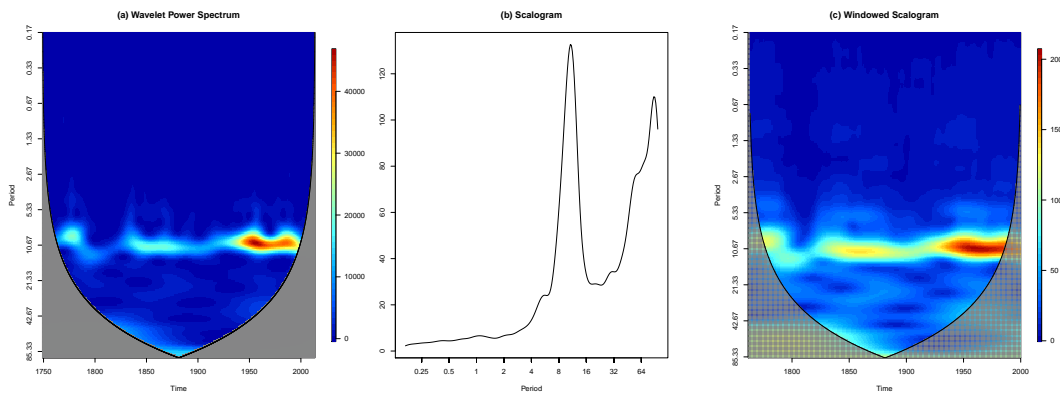


Figure 16: (a) Wavelet power spectrum, (b) scalogram, and (c) windowed scalogram of the sunspots time series, with `energy_density = FALSE`. These plots depict the same as Figure 15, but the bias in favour of large scales is present. Nevertheless, this is recommended for computing the corresponding scale indices.

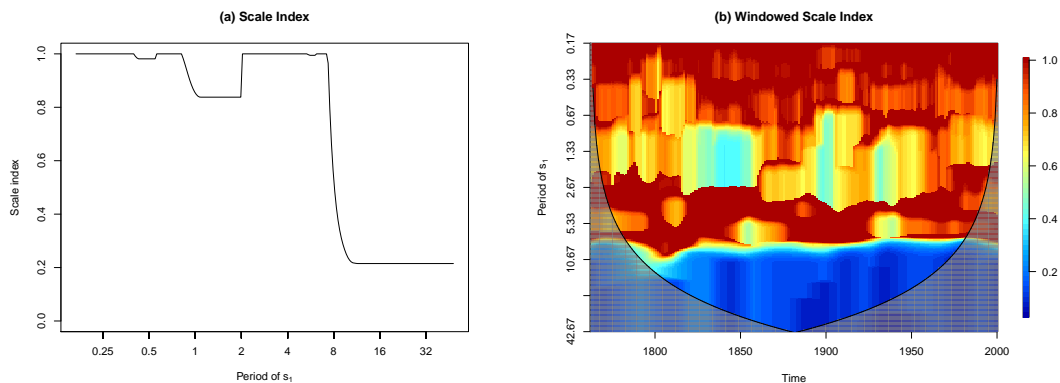


Figure 17: (a) Scale indices and (b) windowed scale indices of the sunspots time series. In these plots, the use of the scale indices to determine the sunspots period is depicted. The period is estimated by the minimum scale s_1 for which the scale indices are stabilized around lower values, presenting the transition from non-periodicity to a far more periodic signal. The windowed scale indices in plot (b) are specially useful for non-stationary time series because they can detect changes in the sunspots period over time.

- & *Mathematics with Applications*, 60(3):634–641, 2010. URL <https://doi.org/10.1016/j.camwa.2010.05.010>. [p164, 170, 174]
- V. J. Bolós and R. Benítez. *wavScalogram: Wavelet Scalogram Tools for Time Series Analysis*, 2021. URL <https://CRAN.R-project.org/package=wavScalogram>. R package version 1.1.1. [p164]
- V. J. Bolós, R. Benítez, R. Ferrer, and R. Jammazi. The windowed scalogram difference: a novel wavelet tool for comparing time series. *Applied Mathematics and Computation*, 312:49–65, 2017. URL <https://doi.org/10.1016/j.amc.2017.05.046>. [p164, 170, 172, 174]
- V. J. Bolós, R. Benítez, and R. Ferrer. A new wavelet tool to quantify non-periodicity of non-stationary economic time series. *Mathematics*, 8:844–859, 2020. URL <https://doi.org/10.3390/math8050844>. [p164, 174, 175, 177, 178]
- R. Carmona and B. Torresani. *Rwave: Time-Frequency Analysis of 1-D Signals*, 2021. URL <https://CRAN.R-project.org/package=Rwave>. R package version 2.6-0. [p166]
- I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1992. ISBN 9780898712742. [p164]
- M. Farge. Wavelet transforms and their applications to turbulence. *Annual Review of Fluid Mechanics*, 24:395–458, 1992. URL <https://doi.org/10.1146/annurev.fl.24.010192.002143>. [p165]
- T. C. Gouhier, A. Grinsted, and V. Simko. *R package biwavelet: Conduct Univariate and Bivariate Wavelet Analyses*, 2021. URL <https://github.com/tgouhier/biwavelet>. (Version 0.20.21). [p164]
- Y. Liu, X. San Liang, and R. H. Weisberg. Rectification of the bias in the wavelet power spectrum. *Journal of Atmospheric and Oceanic Technology*, 24(12):2093–2102, 2007. URL <https://doi.org/10.1175/2007JTECH0511.1>. [p166]
- S. Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Academic Press, 2008. ISBN 978-0-08-092202-7. [p164, 165]
- P. Montero and J. Vilar. TSclust: an R package for time series clustering. *Journal of Statistical Software*, 62(1):1–43, 2014. URL <https://doi.org/10.18637/jss.v062.i01>. [p180]
- G. Nason. *wavethresh: Wavelets Statistics and Transforms*, 2022. URL <https://CRAN.R-project.org/package=wavethresh>. R package version 4.6.9. [p164]
- P. Roebuck and Rice University’s DSP group. *rwf: ‘Rice Wavelet Toolbox’ Wrapper*, 2022. URL <https://CRAN.R-project.org/package=rwf>. R package version 1.0.2. [p164]
- A. Roesch and H. Schmidbauer. *WaveletComp: Computational Wavelet Analysis*, 2018. URL <https://CRAN.R-project.org/package=WaveletComp>. R package version 1.1. [p164]
- D. Savchev and G. Nason. *hwwntest: Tests of White Noise using Wavelets*, 2018. URL <https://CRAN.R-project.org/package=hwwntest>. R package version 1.3.1. [p164]
- S. A. C. Taylor, T. Park, and I. A. Eckley. Multivariate locally stationary wavelet analysis with the mvLSW R package. *Journal of Statistical Software*, 90(11):1–19, 2019. URL <https://doi.org/10.18637/jss.v090.i11>. [p164]
- C. Torrence and G. Compo. A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, 79(1):61–78, 1998. URL [https://doi.org/10.1175/1520-0477\(1998\)079<0061:APGTWA>2.0.CO;2](https://doi.org/10.1175/1520-0477(1998)079<0061:APGTWA>2.0.CO;2). [p165, 166, 168, 173]
- C. Torrence and P. J. Webster. Interdecadal changes in the ENSO–monsoon system. *Journal of Climate*, 12(8):2679–2690, 1999. URL [https://doi.org/10.1175/1520-0442\(1999\)012<2679:ICITEM>2.0.CO;2](https://doi.org/10.1175/1520-0442(1999)012<2679:ICITEM>2.0.CO;2). [p173]
- B. Whitcher. *waveslim: Basic Wavelet Routines for One-, Two-, and Three-Dimensional Signal Processing*, 2020. URL <https://CRAN.R-project.org/package=waveslim>. R package version 1.8.2. [p164]

Vicente J. Bolós
 Department of Business Mathematics. University of Valencia
 Avda. Tarongers s/n. 46022. Valencia
 Spain
vicente.bolos@uv.es

Rafael Benítez
Department of Business Mathematics. University of Valencia
Avda. Tarongers s/n. 46022. Valencia
Spain
rabesua@uv.es

ClusTorus: An R Package for Prediction and Clustering on the Torus by Conformal Prediction

by Seungki Hong and Sungkyu Jung

Abstract Protein structure data consist of several dihedral angles, lying on a multidimensional torus. Analyzing such data has been and continues to be key in understanding functional properties of proteins. However, most of the existing statistical methods assume that data are on Euclidean spaces, and thus they are improper to deal with angular data. In this paper, we introduce the package **ClusTorus** specialized to analyzing multivariate angular data. The package collects some tools and routines to perform algorithmic clustering and model-based clustering for data on the torus. In particular, the package enables the construction of conformal prediction sets and predictive clustering, based on kernel density estimates and mixture model estimates. A novel hyperparameter selection strategy for predictive clustering is also implemented, with improved stability and computational efficiency. We demonstrate the use of the package in clustering protein dihedral angles from two real data sets.

1 Introduction

Multivariate angular or circular data have found applications in some research domains including geology (e.g., paleomagnetic directions) and bioinformatics (e.g., protein dihedral angles). Due to the cyclic nature of angles, usual vector-based statistical methods are not directly applicable to such data. A p -variate angle $\theta = (\theta_1, \dots, \theta_p)^T$ lies on the p -dimensional torus $\mathbb{T}^p = [0, 2\pi)^p$ in which the angles 0 and 2π are identified as the same point. Likewise, angles θ and $\theta \pm 2\pi$ are the same data point on the torus. Thus, statistical models and predictions on the torus should reflect this geometric constraint.

A prominent example in which multivariate angular data appear is the analysis of protein structures. As described in [Branden and Tooze \(1999\)](#), the functional properties of proteins are determined by the ordered sequences of amino acids and their spatial structures. These structures are determined by several dihedral angles, and thus, protein structures are commonly described on multidimensional tori. The p -dimensional torus \mathbb{T}^p is the sample space we consider in this paper. Especially, for the 2-dimensional case, the backbone chain angles ϕ, ψ of a protein are commonly visualized by the Ramachandran plot, a scatter plot of dihedral angles in a 2-dimensional flattened torus \mathbb{T}^2 ([Lovell et al., 2003](#); [Oberholser, 2010](#)). In [Figure 1](#), several clustering results are visualized on the Ramachandran plot for the protein angles of SARS-CoV-2 virus, which caused the 2020-2021 pandemic ([Coronaviridae Study Group of the International Committee on Taxonomy of Viruses. et al., 2020](#)). Since the structures in protein angles are related to functions of the protein, it is of interest to analyze the scatter of the angles through, for example, density estimation and clustering. Note that the protein structure data are routinely collected and publicly available at Protein Data Bank ([Berman et al., 2003](#)) and importing such data into R is made easy by the package [bio3d](#) ([Grant et al., 2006, 2021](#)).

We introduce the R package [ClusTorus](#) ([Jung and Hong, 2021](#)) which provides various tools for handling and clustering multivariate angular data on the torus. The package provides angular adaptations of usual clustering methods such as the k -means clustering and pairwise angular distances, which can be used as an input for distance-based clustering algorithms, and implements a novel clustering method based on conformal prediction framework ([Vovk et al., 2005](#)). Also implemented in the package are the EM algorithms and an elliptical k -means algorithm for fitting mixture models on the torus, and a kernel density estimation. We will introduce various clustering tools implemented in the package, explaining choices in conformal prediction using two sets of example data. We also present the theoretical and technical background, and demonstrate these tools with R codes.

For data on the torus, there are a few previous works for mixture modeling and clustering. [Mardia et al. \(2007\)](#) proposed a mixture of bivariate von Mises distributions for data on \mathbb{T}^2 , with an application to modeling protein backbone chain angles. [Mardia et al. \(2012\)](#) proposed a density estimation on the torus, based on a mixture of approximated von Mises sine distributions, for higher dimensional cases, but the proposed EM algorithm tends to be unstable when sample sizes are limited. The R package [BAMBI](#) ([Chakraborty and Wong, 2019, 2020](#)) provides routines to fit such von Mises mixture models using MCMC, but is only applicable to bivariate (and univariate) angles in \mathbb{T}^2 . We have implemented EM algorithms (for $p = 2$) and the elliptical k -means algorithm (for any p), originally proposed for vector-valued data ([Sung and Poggio, 1998](#); [Bishop, 2006](#); [Shin et al., 2019](#)), for fitting mixture models on the torus. To the best of authors' knowledge, **ClusTorus** is the first implementation of methods for

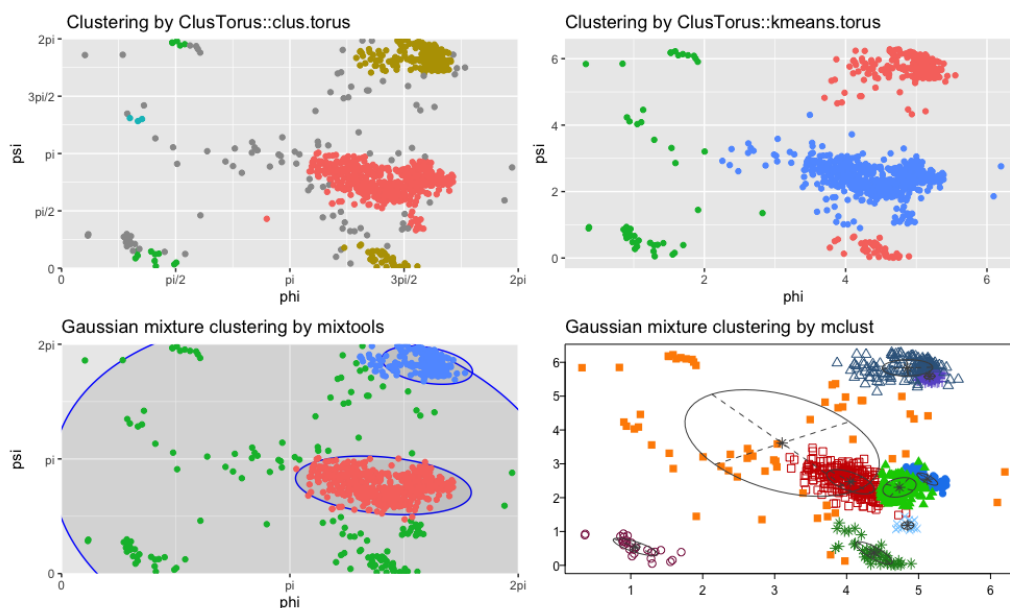


Figure 1: Several clustering results on Ramachandran plot for SARS-CoV-2 by using `clus.torus` (top left) and `kmeans.torus` (top right), both implemented in **ClusTorus**, `mixtools::mvnormalmixEM` (bottom left), in which the number of components 3 is prespecified, and `mclust::Mclust` (bottom right), in which the number of components is chosen by BIC. Gray points in the top-left panel are “outliers”, automatically assigned by `clus.torus`.

fitting mixture models on multidimensional tori of any dimension.

Algorithmic clustering for data on the torus has also been proposed. For example, Gao et al. (2018) used an extrinsic k -means algorithm for clustering protein angles. While this algorithm is not always satisfactory, it is implemented in **ClusTorus** for a quick-and-dirty analysis. The top right panel of Figure 1 depicts the result of applying this algorithm with $k = 3$. Note that the popular R packages `mixtools` (Benaglia et al., 2009) and `mclust` (Scrucca et al., 2016) provide misleading clustering results, when applied to data on the torus. As we illustrate in Figure 1, these tools do not take into account the cyclic nature of the angular data.

The main contribution of **ClusTorus** is an implementation of the predictive clustering approaches of Jung et al. (2021) and Shin et al. (2019). For this, the conformal prediction framework of Vovk et al. (2005) is extended for multivariate angular data. The conformal prediction is a distribution-free method of constructing prediction sets, and our implementation uses kernel density estimates and mixture models, both based on the multivariate von Mises distribution (Mardia et al., 2012). Furthermore, by using Gaussian-like approximations of the von Mises distributions and a graph-theoretic approach, flexible clusters, composed of unions of ellipsoids on \mathbb{T}^p , can be identified. The proposed predictive clustering can be obtained by simply using `clus.torus` as follows.

```
library(ClusTorus)
set.seed(2021)
ex <- clus.torus(SARS_CoV_2)
plot(ex)
```

The result of the predictive clustering is visualized in the top left panel of Figure 1, which is generated by `plot(ex)`. The dataset `SARS_CoV_2`, included in **ClusTorus**, collects the dihedral angles ϕ, ψ in the backbone chain B of SARS-CoV-2 spike glycoprotein. The raw coronavirus protein data are available at Protein Data Bank with id 6VXX (Walls et al., 2020), and can be retrieved by using R package `bio3d`. The function `clus.torus` performs three core procedures—conformal prediction, hyperparameter selection and cluster assignment—for predictive clustering.

The rest of this article focuses on introducing the three core procedures: (i) the conformal prediction framework, including our choices of the conformity scores, (ii) hyperparameter selection and (iii) cluster assignment. After demonstrating how the package **ClusTorus** can be used for clustering of \mathbb{T}^2 - and \mathbb{T}^4 -valued data, we describe how the main function `clus.torus` and other clustering algorithms such as k -means and hierarchical clustering can be used to analyze data on the torus. In the Appendix, we provide technical details and options in fitting mixture models on the torus, and a list of S3 classes defined in **ClusTorus**.

2 Conformal prediction

The conformal prediction framework (Vovk et al., 2005) is one of the main ingredients of our development. Based on the work of Vovk et al. (2005) and Lei et al. (2013, 2015), we briefly introduce the basic concepts and properties of conformal prediction. Suppose that we observe a sample of size n , $X_i \sim F$ where $X_i \in \mathbb{T}^p$ for each i and that the sequence $\mathbb{X}_n = \{X_1, \dots, X_n\}$ is exchangeable. Then, for a new $X_{n+1} \sim F$, the prediction set $C_n = C_n(\mathbb{X}_n)$ is said to be valid at level $1 - \alpha$ if:

$$P(X_{n+1} \in C_n) \geq 1 - \alpha, \quad \alpha \in (0, 1), \tag{1}$$

where P is the corresponding probability measure for $\mathbb{X}_{n+1} = \mathbb{X}_n \cup \{X_{n+1}\}$.

For a given $x \in \mathbb{T}^p$, write $\mathbb{X}_n(x) = \mathbb{X}_n \cup \{x\}$. Consider the null hypothesis $H_0 : X_{n+1} = x$, where $X_{n+1} \sim F$. To test the hypothesis, the conformal prediction framework uses *conformity scores* σ_i defined as follows:

$$\begin{aligned} \sigma_i(x) &:= g(X_i, \mathbb{X}_n(x)), \quad \forall i = 1, \dots, n+1, \\ \sigma(x) &:= g(x, \mathbb{X}_n(x)) = \sigma_{n+1}(x), \end{aligned}$$

for some real valued function g , which measures the conformity or similarity of a point to the given set. If $X_{(1)}, \dots, X_{(n+1)}$ are ordered to satisfy $\sigma_{(1)} \leq \dots \leq \sigma_{(n+1)}$ for $\sigma_{(i)} = g(X_{(i)}, \mathbb{X}_{n+1})$, then we may say that $X_{(n+1)}$ is the most similar point to \mathbb{X}_{n+1} .

Consider the following quantity:

$$\pi(x) = \frac{1}{n+1} \sum_{i=1}^{n+1} I(\sigma_i(x) \leq \sigma_{n+1}(x)), \quad I(A) = \begin{cases} 1, & A \text{ is true,} \\ 0, & \text{otherwise,} \end{cases}$$

which can be understood as a p-value for the null hypothesis H_0 . The *conformal prediction set* of level $1 - \alpha$ is constructed as

$$C_n^\alpha = \{x : \pi(x) > \alpha\}. \tag{2}$$

Because the sequence $\mathbb{X}_n(x)$ is exchangeable under H_0 , $\pi(x)$ is uniformly distributed on $\left\{\frac{1}{n+1}, \dots, 1\right\}$. With this property, it can be shown that the conformal prediction set is valid for finite samples, i.e., (1) holds with C_n replaced by C_n^α for any F , that is, the prediction set is distribution-free (Lei et al., 2013). The performance of the conformal prediction highly depends on the choice of conformity score σ . In some previous works on conformal prediction (Lei et al., 2013, 2015; Shin et al., 2019; Jung et al., 2021), the quality of prediction sets using density based conformity scores has been satisfactory.

We demonstrate a construction of the conformal prediction set with a kernel density estimate-based conformity score, defined later in (3), for the data shown in Figure 1. With the conformity score given by (3), `cp.torus.kde` computes the conformal prediction set C_n^α at a given level $1 - \alpha$ ($\alpha = 0.1$ below), by performing the kernel density estimation. The function tests the inclusion in C_n^α of each point (ϕ, ψ) over a fine grid of \mathbb{T}^2 , and the result of the testing is shown as the boolean indices in the column `Cn` of the output below. The columns `Lminus` and `Lplus` provide approximated prediction sets, defined in Jung et al. (2021).

```
cp.kde <- cp.torus.kde(SARS_CoV_2)
cp.kde
```

Conformal prediction sets (Lminus, Cn, Lplus) based on kde with concentration 25

Testing inclusion to the conformal prediction set with level = 0.1 :

```
-----
      phi psi Lminus   Cn Lplus level
1  0.00000000  0 FALSE FALSE FALSE  0.1
2  0.06346652  0 FALSE FALSE FALSE  0.1
3  0.12693304  0 FALSE FALSE FALSE  0.1
4  0.19039955  0 FALSE FALSE FALSE  0.1
5  0.25386607  0 FALSE FALSE FALSE  0.1
6  0.31733259  0 FALSE FALSE FALSE  0.1
7  0.38079911  0 FALSE FALSE FALSE  0.1
8  0.44426563  0 FALSE FALSE FALSE  0.1
9  0.50773215  0 FALSE FALSE FALSE  0.1
10 0.57119866  0 FALSE FALSE FALSE  0.1
```

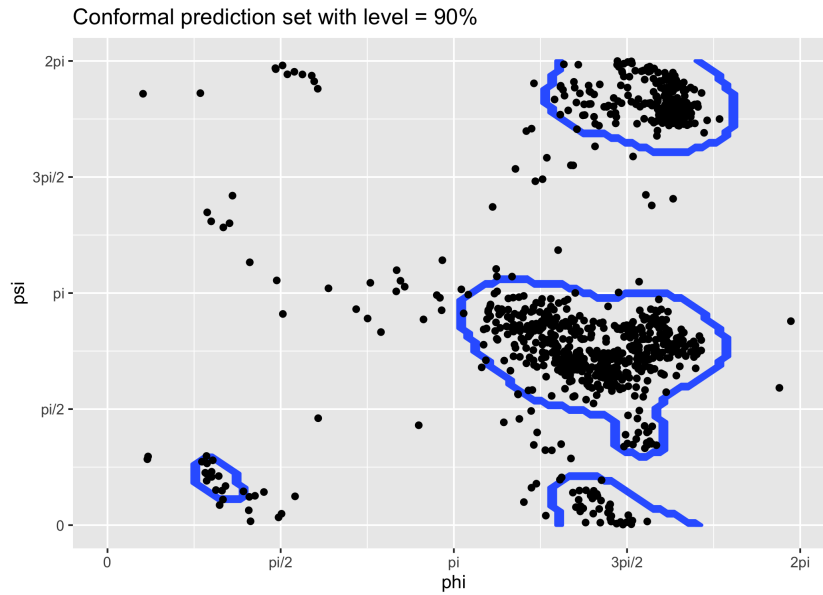


Figure 2: The Ramachandran plot for SARS-CoV-2, with boundaries of the conformal prediction set, whose conformity score is (3) with $\kappa = 25$ for level $1 - \alpha = 0.9$.

9990 rows are omitted.

The concentration parameter κ of the kernel density estimation and the level(s) of the prediction set can be designated by providing arguments concentration and level. By default, these values are set as concentration = 25 and level = 0.1.

The output `cp.kde` is an S3 object with class `cp.torus.kde`, for which a generic method `plot` is available. The conformal prediction set for SARS_CoV_2 data can be displayed on the Ramachandran plot, as follows. The result is shown in Figure 2.

```
plot(cp.kde)
```

Inductive conformal prediction

If the sample size n and the number N of grid points over \mathbb{T}^p are large, evaluating $n + N$ conformity scores may take a long time. That is, constructing the conformal prediction set suffers from high computational costs. A workaround for this inefficiency is *inductive conformal prediction*, which enjoys significantly lower computational cost. The inductive conformal prediction framework is based on splitting the data into two sets. The algorithm for inductive conformal prediction is given in Algorithm 1.

Algorithm 1 Inductive Conformal Prediction

- 1: **procedure** INDUCTIVE CONFORMAL PREDICTION($\{X_1, \dots, X_n\}, \alpha, n_1 < n$)
 - 2: Split the data randomly into $\mathbb{X}_1 = \{X_1, \dots, X_{n_1}\}, \mathbb{X}_2 = \{X_{n_1+1}, \dots, X_n\}$.
 - 3: Construct σ with $\sigma(x) = g(x, \mathbb{X}_1)$ for some function g .
 - 4: Put $\sigma_i = g(X_{n_1+i}, \mathbb{X}_1)$ and order as $\sigma_{(1)} \leq \dots \leq \sigma_{(n_2)}$, where $n_2 = n - n_1$.
 - 5: Construct $\hat{C}_n^\alpha = \{x : \sigma(x) \geq \sigma_{(i_{n,\alpha})}\}$ where $i_{n,\alpha} = \lfloor (n+1)\alpha \rfloor$.
 - 6: **end procedure**
-

While the sizes n_1 and n_2 of two split data sets can be of any size, they are typically set as equal sizes. It is well-known that the output \hat{C}_n^α of the algorithm also satisfies the distribution-free finite-sample validity (Vovk et al., 2005; Lei et al., 2015). For fast computation, the inductive conformal prediction is primarily used in constructing prediction sets and clustering, in our implementation of **ClusTorus**. Specifically, `icp.torus` implements Algorithm 1 for several prespecified conformity scores. As already mentioned, we need to choose the conformity score σ carefully for better clustering performances.

Before we discuss our choices of the conformity scores, we first illustrate how the functions in **ClusTorus** are used to produce inductive conformal prediction sets. The following codes show a calculation of the inductive conformal prediction set for the data SARS_CoV_2. The conformal prediction set with the conformity score given by kernel density estimates (3) can be constructed by `icp.torus` and `icp.torus.eval`. The function `icp.torus` computes σ_i 's in line 4 of Algorithm 1 and `icp.torus.eval` tests whether pre-specified evaluation points are included in \hat{C}_n^α . If these evaluation points are not supplied, then `icp.torus.eval` creates a grid of size 100×100 (for $p = 2$).

```
set.seed(2021)
icp.torus.kde <- icp.torus(SARS_CoV_2, model = "kde", concentration = 25)
icp.kde <- icp.torus.eval(icp.torus.kde, level = 0.1)
icp.kde
```

Conformal prediction set (Chat_kde)

Testing inclusion to the conformal prediction set with level = 0.1:

```
-----
          X1 X2 inclusion
1  0.00000000 0    FALSE
2  0.06346652 0    FALSE
3  0.12693304 0    FALSE
4  0.19039955 0    FALSE
5  0.25386607 0    FALSE
6  0.31733259 0    FALSE
7  0.38079911 0    FALSE
8  0.44426563 0    FALSE
9  0.50773215 0    FALSE
10 0.57119866 0    FALSE
```

9990 rows are omitted.

In the codes above, the data splitting for `icp.torus` is done internally, and can be inspected by `icp.torus.kde$split.id`.

We now introduce our choices for the conformity score σ in the next two subsections.

Conformity score from kernel density estimates

For the 2-dimensional case, Jung et al. (2021) proposed to use the kernel density estimate based on the von Mises kernel (Marzio et al., 2011) for the conformity score. A natural extension to the p -dimensional tori, for $p \geq 2$, is

$$g(u, \mathbb{X}_n(x)) = \frac{1}{n+1} \sum_{i=1}^{n+1} K_\kappa(u - X_i), \quad K_\kappa(v) = \prod_{i=1}^p \frac{e^{\kappa \cos(v_i)}}{2\pi I_0(\kappa)}, \quad v = (v_1, \dots, v_p)^T \in \mathbb{T}^p \quad (3)$$

where I_0 is the modified Bessel function of the first kind of order 0, and κ is a prespecified concentration parameter. The function `kde.torus` provides the multivariate von Mises kernel density estimation. For conformal prediction, we take $\sigma(x_i) = g(x_i, \mathbb{X}_n(x))$, and for inductive conformal prediction, we take $\sigma(x) = g(x, \mathbb{X}_1)$.

Conformity scores from mixtures of multivariate von Mises

Our next choices of conformity scores are based on mixture models. Since the multivariate normal distributions are not defined on \mathbb{T}^p , we instead use the multivariate von Mises distribution (Mardia et al., 2008), whose density on \mathbb{T}^p is

$$f(y; \mu, \kappa, \Lambda) = C(\kappa, \Lambda) \exp \left\{ -\frac{1}{2} \left[\kappa^T (2 - 2c(y, \mu)) + s(y, \mu)^T \Lambda s(y, \mu) \right] \right\} \quad (4)$$

where $y = (y_1, \dots, y_p)^T \in \mathbb{T}^p$, $\mu = (\mu_1, \dots, \mu_p)^T \in \mathbb{T}^p$, $\kappa = (\kappa_1, \dots, \kappa_p)^T \in (0, \infty)^p$, $\Lambda = (\lambda_{j,l})$ for $1 \leq j, l \leq p$, $-\infty < \lambda_{jl} < \infty$,

$$\begin{aligned} c(y, \mu) &= (\cos(y_1 - \mu_1), \dots, \cos(y_p - \mu_p))^T, \\ s(y, \mu) &= (\sin(y_1 - \mu_1), \dots, \sin(y_p - \mu_p))^T, \\ (\Lambda)_{jl} &= \lambda_{jl} = \lambda_{lj}, \quad j \neq l, \quad (\Lambda)_{jj} = \lambda_{jj} = 0, \end{aligned}$$

and for some normalizing constant $C(\kappa, \Lambda) > 0$. We write $f(y; \theta) = f(y; \mu, \kappa, \Lambda)$ for $\theta = (\mu, \kappa, \Lambda)$.

For any positive integer J and a mixing probability $\pi = (\pi_1, \dots, \pi_J)$, consider a J -mixture model:

$$p(u; \pi, \theta) = \sum_{j=1}^J \pi_j f(u; \theta_j) \tag{5}$$

where $\theta = (\theta_1, \dots, \theta_J)$, $\theta_j = (\mu_j, \kappa_j, \Lambda_j)$ for $j = 1, \dots, J$. Let $(\hat{\pi}, \hat{\theta})$ be appropriate estimators of (π, θ) based on \mathbb{X}_1 . The plug-in density estimate based on (5) is then

$$p(\cdot; \hat{\pi}, \hat{\theta}) = \sum_{j=1}^J \hat{\pi}_j f(\cdot; \hat{\theta}_j), \tag{6}$$

which can be used as a conformity score by setting $g(\cdot, \mathbb{X}_1) = \hat{p}(\cdot)$. Assuming high concentrations, an alternative conformity score can be set as $g(\cdot, \mathbb{X}_1) = p^{max}(\cdot, \hat{\pi}, \hat{\theta})$ where

$$p^{max}(u; \hat{\pi}, \hat{\theta}) := \max_{j=1, \dots, J} (\hat{\pi}_j f(u; \hat{\theta}_j)) \approx p(u; \hat{\pi}, \hat{\theta}). \tag{7}$$

On the other hand, [Mardia et al. \(2012\)](#) introduced an approximated density function f^* for the p -variate von Mises sine distribution (4) for sufficiently high concentrations and when $\Sigma \succ 0$:

$$f^*(y; \mu, \Sigma) = (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} \left[\kappa^T (2 - 2c(y, \mu)) + s(y, \mu)^T \Lambda s(y, \mu) \right] \right\}$$

where $(\Sigma^{-1})_{jl} = \lambda_{jl}$, $(\Sigma^{-1})_{jj} = \kappa_j$, $j \neq l$. By further approximating via $\theta \approx \sin \theta$, $1 - \frac{\theta^2}{2} \approx \cos \theta$, we write

$$f^*(y; \mu, \Sigma) \approx (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} \left[(y \ominus \mu)^T \Sigma^{-1} (y \ominus \mu) \right] \right\}, \tag{8}$$

where the angular subtraction \ominus stands for

$$X \ominus Y := \left(\arg \left(e^{i(\phi_{x1} - \phi_{y1})} \right), \dots, \arg \left(e^{i(\phi_{xp} - \phi_{yp})} \right) \right)^T,$$

for $X = (\phi_{x1}, \dots, \phi_{xp})^T \in \mathbb{T}^p$ and $Y = (\phi_{y1}, \dots, \phi_{yp})^T \in \mathbb{T}^p$ as defined in [Jung et al. \(2021\)](#) for $p = 2$. By replacing the von Mises density f in (7) with the approximate normal density (8), $\log(p^{max}(\cdot; \pi, \theta))$ is approximated by

$$\begin{aligned} \log(p^{max}(u; \pi, \theta)) &\approx \frac{1}{2} \max_j e(u; \pi_j, \theta_j) + c, \\ e(u; \pi_j, \theta_j) &= - \left(u \ominus \mu_j \right)^T \Sigma_j^{-1} \left(u \ominus \mu_j \right) + 2 \log \pi_j - \log |\Sigma_j| \end{aligned} \tag{9}$$

where $\theta_j = (\mu_j, \Sigma_j)$, $\mu_j = (\mu_{1j}, \dots, \mu_{pj})^T \in \mathbb{T}^p$, $\Sigma_j \in \mathbb{R}^{p \times p}$ and a constant $c \in \mathbb{R}$. Our last choice of the conformity score is

$$g(\cdot, \mathbb{X}_1) = \max_j e(\cdot, \hat{\pi}_j, \hat{\theta}_j). \tag{10}$$

Note that with this choice of conformity score, the conformal prediction set can be expressed as the union of ellipsoids on the torus. That is, the following equalities are satisfied ([Shin et al., 2019](#); [Jung](#)

et al., 2021): Let C_n^e be the level $1 - \alpha$ prediction set using (10). Then

$$\begin{aligned}
 C_n^e &:= \left\{ x \in \mathbb{T}^p : g(x, \mathbb{X}_1) \geq g\left(X_{(i_{n_2, \alpha})}, \mathbb{X}_1\right) \right\} \\
 &= \bigcup_{j=1}^J \hat{E}_j\left(\sigma_{(i_{n_2, \alpha})}\right)
 \end{aligned} \tag{11}$$

where $\hat{E}_j(t) = \left\{ x \in \mathbb{T}^p : (x \ominus \hat{\mu}_j)^T \hat{\Sigma}_j^{-1} (x \ominus \hat{\mu}_j) \leq 2 \log \hat{\tau}_j - \log |\hat{\Sigma}_j| - t \right\}$ for $t \in \mathbb{R}$. Note that $\hat{E}_j(t)$ is automatically vanished if $t \geq 2 \log \hat{\tau}_j - \log |\hat{\Sigma}_j|$.

Implementation

We have implemented four conformity scores, described in the previous section. These are based on

1. kernel density estimate (3),
2. mixture model (6),
3. max-mixture model (7), and
4. ellipsoids obtained by approximating the max-mixture (10).

The function `icp.torus` in **ClusTorus** computes these conformity scores using the inductive conformal prediction framework, and returns `icp.torus` object(s). Table 1 illustrates several important arguments of the function `icp.torus`.

Arguments	Descriptions
<code>data</code>	$n \times d$ matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$
<code>model</code>	A string. One of "kde", "mixture", and "kmeans" which determines the model or estimation methods. If "kde", the model is based on the kernel density estimates. It supports the kde-based conformity score only. If "mixture", the model is based on the von Mises mixture, fitted with an EM algorithm. It supports the von Mises mixture and its variants based conformity scores. If "kmeans", the model is also based on the von Mises mixture, but the parameter estimation is implemented with the elliptical k-means algorithm illustrated in Appendix. It supports the log-max-mixture based conformity score only. If the dimension of data space is greater than 2, only "kmeans" is supported. Default is <code>model = "kmeans"</code> .
<code>J</code>	A scalar or numeric vector for the number(s) of components for <code>model = c("mixture", "kmeans")</code> . Default is <code>J = 4</code> .
<code>concentration</code>	A scalar or numeric vector for the concentration parameter(s) for <code>model = "kde"</code> . Default is <code>concentration = 25</code> .

Table 1: Key arguments and descriptions for the function `icp.torus`

The argument `model` of the function `icp.torus` indicates which conformity score is used. By setting `model = "kde"`, the kde-based conformity score (3) is used. By setting `model = "mixture"` the mixture model (6) is estimated by an EM algorithm, and conformity scores based on (6), (7), (10) are all provided. Setting `model = "kmeans"` provides a mixture model fit by the elliptical k -means algorithm and conformity score based on (10).

The arguments `J` and `concentration` specify the model fitting hyperparameters. To compute conformity scores based on kernel density estimate (3), one needs to specify the concentration parameter κ . Likewise, the number of mixtures, J , needs to be specified in order to fit the mixture model (6) and the variants (7) and (10). The function `icp.torus` takes either a single value (e.g., `J = 4` is the default), or a vector (e.g., `J = 4:30` or `concentration = c(25, 50)`) for arguments `J` and `concentration`. If `J` (or `concentration`) is a scalar, then `icp.torus` returns an `icp.torus` object.

On the other hand, if `J` (or `concentration`) is a numeric vector containing at least two values, then `icp.torus` returns a list of `icp.torus` objects, one for each value in `J` (or `concentration`, respectively). Typically, the hyperparameter J (or κ) is not predetermined, and one needs to choose among a set of candidates. A list of `icp.torus` objects, evaluated for each candidate in vector-valued `J` (or `concentration`) is required for our hyperparameter selection procedure, discussed in a later section.

Let us present an R code example for creating an `icp.torus` object, fitted with `model = "kmeans"` (the default value for argument `model`) and `J = 12`.

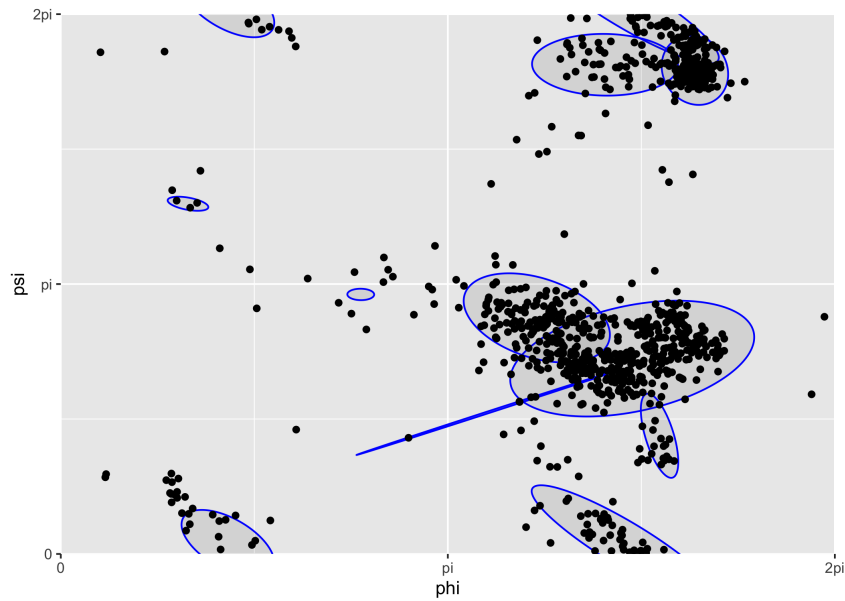


Figure 3: The Ramachandran plot for SARS-CoV-2, with conformal prediction set whose conformity score is (10) with $J = 12$ for level $\alpha = 0.1111$. The plot demonstrates the union of ellipses as (11).

```
set.seed(2021)
icp.torus.12 <- icp.torus(SARS_CoV_2, J = 12)
plot(icp.torus.12, level = 0.1111)
```

The `icp.torus` object has an S3 method `plot`, and the R code `plot(icp.torus.12, level = 0.1111)` plots the ellipses in (11) with α specified by argument `level = 0.1111`. The union of these ellipses is in fact the inductive conformal prediction set of level $1 - \alpha$. The boundaries of the inductive conformal prediction set can be displayed by specifying `ellipse = FALSE`, as follows.

```
plot(icp.torus.12, ellipse = FALSE)
```

The resulting graphic is omitted.

Conformity scores based on mixture model and its variants need appropriate estimators of the parameters, π and θ . If the parameters are poorly estimated, the conformal prediction sets will be constructed trivially and thus become useless. We have implemented two methods of estimation: EM algorithms and the elliptical k -means algorithm, also known as the generalized Lloyd's algorithm (Sung and Poggio, 1998; Bishop, 2006; Shin et al., 2019). EM algorithms for the mixture model (6) are described in Jung et al. (2021), for the 2-dimensional case. Since the EM estimates require long computation time and large sample sizes, extensions to higher-dimensional tori do not seem to apt. The EM estimates of the mixture model parameters can be naturally used for the case of max-mixture (7) and ellipsoids (10) as well. The argument `model = "mixture"` of `icp.torus` works only for the 2-dimensional case. On the other hand, the elliptical k -means algorithm converges much faster even for moderately high-dimensional tori. The elliptical k -means algorithm is used for estimating parameters in the approximated normal density (8), and for computation of the conformity score of ellipsoids (10). The elliptical k -means algorithms for data on the torus are further discussed in the Appendix.

Table 2 summarizes the four choices of conformity scores in terms of model-fitting methods, dimensionality of the data space, and whether clustering is available. Our predictive clustering is implemented only based on the "ellipsoids" conformity score (10). The rationale for this choice is due to the relatively simple form of prediction sets (a union of ellipsoids (11)).

3 Clustering by conformal prediction

We now describe our clustering strategies using the conformal prediction sets. Suppose for now that the level α and the hyperparameter J of the prediction set are given. The basic idea of clustering is to take each connected component of the prediction set as a cluster. For this, we need an algorithm identifying connected components from any prediction set. Since the prediction sets are in general of irregular shapes, such an identification is a quite difficult task. However, as shown in Jung et al. (2021), if the conformal prediction set is of the form (11), clusters are identified by testing the intersection

Conformity Scores	EM	k-means	dim = 2	dim > 2	Clustering
Kernel density (3)			✓		
Mixture (6)	✓		✓		
Max-mixture (7)	✓		✓		
Ellipsoids (10)	✓	✓	✓	✓	✓

Table 2: Conformity scores against available fitting methods, dimensions of the torus, and whether cluster assignment is available.



Figure 4: 1

of ellipsoids. Suppose $C_n^e = \cup_{j=1}^J \hat{E}_j$ where each \hat{E}_j is an ellipsoid. Let the (i, j) th entry of a square matrix A be 0 if $\hat{E}_i \cap \hat{E}_j = \emptyset$, 1 otherwise. Then, A is the adjacent matrix of a graph whose nodes and edges represent the ellipsoids and intersections, respectively. The adjacent matrix A gives a partition $I_1, \dots, I_K \subseteq \{1, \dots, J\}$ satisfying

$$\hat{E}_{i_k} \cap \hat{E}_{i_{k'}} = \emptyset, \quad k \neq k'$$

where $1 \leq k, k' \leq K, i_k \in I_k, i_{k'} \in I_{k'}$. This implies that the union of ellipsoids, $U_k = \cup_{i \in I_k} \hat{E}_i$, whose indices are in a connected component I_k for some k , can be regarded as a cluster. That is, U_1, \dots, U_K are the disjoint clusters. With this, the conformal prediction set naturally generates K clusters. Note that testing the intersection of ellipsoids can be done efficiently (which is a univariate root finding problem (Gilitschenski and Hanebeck, 2012)), while testing the intersection of arbitrarily shaped sets is not feasible in general. This is the reason why we only use the conformity score of the form (10), the prediction set from which is exactly the union of ellipsoids.

We now describe how the cluster labels are assigned to data points. Each data point included in the prediction set is automatically assigned to the cluster which contains the point. For the data points which are not included in the conformal prediction set, we have implemented two different types for cluster assignment, as defined in Jung et al. (2021). The first is to assign the *closest* cluster label. The notion of closest cluster can be defined either by the Mahalanobis distance $(x \ominus \hat{\mu}_j)^T \hat{\Sigma}_j^{-1} (x \ominus \hat{\mu}_j)$, the approximate log-density (9), or the largest posterior probability $\hat{P}(Y = k | X = x)$. For example, for $x \notin C_n^e$, let E_i be the set with the largest approximate log-density $\hat{e}_i(x)$. If $i \in I_k$, then x is assigned to the cluster k . These provide three choices of cluster assignment, depending on the definition of “closeness.” The last choice is to regard the excluded points as outliers. That is, if $x \notin C_n^e$, then the point x is labeled as “outlier.” This outlier-disposing clustering may be more appropriate for the cases where some of data points are truly outliers. Figure 4 compares the two different types of clustering assignment.

The function `cluster.assign.torus`, which takes as input an `icp.torus` object and level α , generates the clusters as we described above. The output of the function is an S3 object with class `cluter.obj`, and includes the cluster memberships of all data points, for each and every cluster assignment method we discussed above. The output of `cluster.assign.torus` includes the number of clusters detected, the cluster assignment results for the first 10 observations, and cluster sizes, as shown in the code example below.

```
c <- cluster.assign.torus(icp.torus.12, level = 0.1111)
```

c

Number of clusters: 5

Clustering results by log density:

[1] 1 1 1 1 2 4 2 1 3 1

cluster sizes: 538 372 39 4 19

Clustering results by posterior:

[1] 5 5 5 5 2 4 3 5 3 5

cluster sizes: 6 310 104 4 548

Clustering results by representing outliers:

[1] 1 1 1 1 2 6 2 1 6 1

cluster sizes: 508 343 15 3 0 103

Note: cluster id = 6 represents outliers.

Clustering results by Mahalanobis distance:

[1] 1 1 1 1 2 4 2 1 3 1

cluster sizes: 533 372 39 4 24

962 clustering results are omitted.

The clustering results contained in the object c can be visualized as follows.

```
plot(c, assignment = "log.density")
```

```
plot(c, assignment = "outlier")
```

The results are displayed in Figure 4. When the argument assignment is not specified, the outlier disposing assignment is chosen by default.

4 Hyperparameter selection

Poor choices of conformity score result in too wide prediction sets. Thus, we need to choose the hyperparameters elaborately for a better conformal prediction set and for a better clustering performance. The hyperparameters are the concentration parameter κ (for the case (3)) or the number of mixture components J (for the cases (6), (7), (10)), as well as the level α for all cases. There have been some efforts to select the optimal hyperparameters by introducing adequate criteria. [Lei et al. \(2013\)](#) and [Jung et al. \(2021\)](#) each proposed criteria based on the minimum volume of the conformal prediction set. However, as we shall see, these approaches become computationally infeasible for higher dimensions.

We briefly review the criterion used in [Jung et al. \(2021\)](#). Assume for now that mixture models are used; that is, (J, α) are the hyperparameters of interest. For a set $C \subseteq \mathbb{T}^p$, let $\mu(C)$ be the volume of C . Without loss of generality, we can assume that $\mu(\mathbb{T}^p) = 1$. For a given level α , the optimal choice of hyperparameter J minimizes $\mu(C_n(\alpha, J))$ of conformal prediction set $C_n(\alpha, J)$. To choose α and J altogether, [Jung et al. \(2021\)](#) proposed to use the following criterion:

$$(\hat{\alpha}, \hat{J}) = \arg \min_{\alpha, J} \alpha + \mu(C_n(\alpha, J)). \quad (12)$$

Note that if (κ, α) are the hyperparameters, then J and \hat{J} in (12) are replaced by κ and $\hat{\kappa}$.

To evaluate (12), one needs to have a set of candidates for J (or κ), and conformal prediction sets corresponding to each choice of J (or κ , respectively). For this purpose, the function `icp.torus` is designed to take as input a set of hyperparameter candidates. As an example, the following code evaluates the inductive conformal prediction sets for data `SARS_CoV_2`, fitted by mixture models with the number of components given by each $J = 3, 4, \dots, 35$.

```
set.seed(2021)
```

```
icp.torus.objects <- icp.torus(SARS_CoV_2, J = 3:35)
```

The result, `icp.torus.objects`, is a list of 33 `icp.torus` objects. Evaluating [Jung et al. \(2021\)](#)'s criterion (12) is implemented in the function `hyperparam.torus`. There, the criterion (12) is termed "elbow", since the minimizer $(\hat{\alpha}, \hat{J})$ is typically found at an elbow of the graph of the objective function.

```

hyperparam.out <- hyperparam.torus(icp.torus.objects)
hyperparam.out

Type of conformity score: kmeans general
Optimizing method: elbow
-----
Optimally chosen parameters. Number of components = 12 , alpha = 0.1111111
Results based on criterion elbow :
      J   alpha   mu criterion
2241 12 0.1111111 0.1215 0.2326111
2244 12 0.1172840 0.1169 0.2341840
2242 12 0.1131687 0.1211 0.2342687
2243 12 0.1152263 0.1198 0.2350263
2001 11 0.1172840 0.1179 0.2351840
2240 12 0.1090535 0.1265 0.2355535
2245 12 0.1193416 0.1169 0.2362416
2002 11 0.1193416 0.1175 0.2368416
2494 13 0.1316872 0.1053 0.2369872
2004 11 0.1234568 0.1136 0.2370568
2246 12 0.1213992 0.1161 0.2374992
2003 11 0.1213992 0.1163 0.2376992
2005 11 0.1255144 0.1123 0.2378144
1999 11 0.1131687 0.1248 0.2379687
2239 12 0.1069959 0.1310 0.2379959

8004 rows are omitted.

Available components:
[1] "model"      "option"      "results"     "icp.torus"  "Jhat"        "alphahat"

```

It can be checked that the choice of $J = 12$ and $\alpha = 0.1111$ in the previous examples was indeed given by the option "elbow".

In computing the criterion (12), the volume $\mu(C_n(\alpha, J))$ is numerically approximated. This is feasible for data on $\mathbb{T}^2 = [0, 2\pi)^2$ by inspecting the inclusion of each point of a fine grid. However, for high dimensional cases, for example \mathbb{T}^4 , evaluating the volume becomes computationally infeasible. In fact, as the dimension increases, the number of required inspections grows exponentially. Furthermore, the function $(\alpha, J) \rightarrow \alpha + \mu(C_n(\alpha, J))$ is typically not a convex function and has multiple local minima. Thus, the choice of $(\hat{\alpha}, \hat{J})$ by (12) tends to be unstable, resulting in high variability of the clustering results. Therefore, evaluating (12) is not practical for high-dimensional data.

To this end, we have developed and implemented a computationally more efficient procedure for hyperparameter selection, which also provides more stable clustering results. This procedure is a two-step procedure, first choosing the model parameter J , then choosing the level α . The two-step procedure is implemented for choosing J and α , but not for κ and α . Our approach is in contrast to the approaches in [Lei et al. \(2013\)](#) and [Shin et al. \(2019\)](#) in which they only choose the model parameter for a prespecified level α .

The first step of the procedure is to choose J , without making any reference to the level α . Choosing J can be regarded as selecting an appropriate mixture model. The model selection is based on either the (prediction) risk, Akaike information criterion ([Akaike, 1974](#)), or Bayesian information criterion ([Schwarz, 1978](#)). Since the mixture model-based conformity scores (6), (7) and (10) are actually the density or the approximated log-density of the mixture model, we use the conformity scores in place of the likelihood. For example, the sum of the conformity scores (10) over the given data is exactly the fitted log-likelihood. Specifically, let $\mathbb{X}_1, \mathbb{X}_2$ be the splitted datasets given by Algorithm 1 and $\mathbb{X} = \mathbb{X}_1 \cup \mathbb{X}_2$. Let $\sigma(\cdot) = \log g(\cdot; \mathbb{X}_1)$ if g is given by (6) and (7) or $\sigma(\cdot) = g(\cdot; \mathbb{X}_1)$ if g is given by (10). Recall that g is the conformity score, and it depends on the estimated model \hat{p} . Then, the function σ we defined above also depends on the model \hat{p} , and the criterion R can be defined as follows:

$$R(\mathbb{X}, \hat{p}) = \begin{cases} -2 \sum_{x \in \mathbb{X}_2} \sigma(x) & \text{if the criterion is the risk,} \\ -2 \sum_{x \in \mathbb{X}_1} \sigma(x) + 2k & \text{if the criterion is AIC,} \\ -2 \sum_{x \in \mathbb{X}_1} \sigma(x) + k \log n_1 & \text{if the criterion is BIC,} \end{cases}$$

where k is the number of model parameters and n_1 is the cardinality of \mathbb{X}_1 . The function `hyperparam.J` computes the minimizer \hat{J} of the criterion, as summarized in Algorithm 2.

The fitted models $\hat{p}_{j_1}, \dots, \hat{p}_{j_n}$ of Algorithm 2 are exactly the outputs of `icp.torus` for various

Algorithm 2 hyperparam.J

-
- 1: **procedure** HYPERPARAM.J($\mathbb{X} \subset \mathbb{T}^p$, fitted models $\hat{p}_{j_1}, \dots, \hat{p}_{j_n}$, criterion R)
 - 2: Evaluate $R_j = R(\mathbb{X}, \hat{p}_j)$ for $j = j_1, \dots, j_n$.
 - 3: Evaluate $\hat{j} = \arg \min_{j \in \{j_1, \dots, j_n\}} R_j$.
 - 4: Output $\hat{j}, \hat{p}_{\hat{j}}$.
 - 5: **end procedure**
-

$J = j_1, \dots, j_n$. Which criterion to use is specified by setting the argument option of hyperparam.J. The argument option = "risk", "AIC", or "BIC" is for the risk, AIC, or BIC, respectively. By choosing \hat{j} , we also fix the model $\hat{p}_{\hat{j}}$ for the next step.

The second step is to choose the level $\alpha \in (0, 1)$ for the chosen \hat{j} and $\hat{p}_{\hat{j}}$, so that the clustering result is stable over perturbations of α . If the number of clusters does not change by varying the level $\alpha \in I$ for some interval I , we regard that the clustering result is stable on I . If I is sufficiently wide, it is reasonable to choose an $\alpha \in I$. Thus, our strategy is to find the most wide interval $I = [a, b] \subseteq (0, 1)$ whose elements construct the same number of clusters, and to set $\hat{\alpha}$ as the midpoint of the interval, i.e. $\hat{\alpha} = (a + b)/2$. However, choosing α large, e.g. $\alpha > 0.5$, results in a too small coverage $1 - \alpha$ of the prediction set. Thus, we restrict the searching area as $[0, M]$ for $M \in (0, 1)$ which is close to 0, and find the desirable I in the restricted area $[0, M]$ rather than the whole interval $[0, 1]$. This strategy is implemented in hyperparam.alpha, and the algorithm is described in Algorithm 3.

Algorithm 3 hyperparam.alpha

-
- 1: **procedure** HYPERPARAM.ALPHA(fitted model \hat{p} , $n_2 := |\mathbb{X}_2|$, $M \in [0, 1]$)
 - 2: Evaluate the number of clusters c_{α_j} for $\alpha_j = j/n_2$, $j = 1, \dots, \lfloor n_2 M \rfloor$.
 - 3: Set $A = \{j : c_{\alpha_{j-1}} \neq c_{\alpha_j}, j = 2, \dots, \lfloor n_2 M \rfloor\}$.
 - 4: For $A = \{\alpha_{j_1}, \dots, \alpha_{j_N}\}$ find $i = \arg \max_{k \in \{1, \dots, N-1\}} \alpha_{j_{k+1}} - \alpha_{j_k}$.
 - 5: Output $\hat{\alpha} = (\alpha_{j_{i+1}} + \alpha_{j_i}) / 2$
 - 6: **end procedure**
-

Note that we could alternatively input an array of levels, for the argument alphavec of hyperparam.alpha, if there is a prespecified searching area. In our experience, setting $M = 0.15$ gives generally satisfying results. By setting $M = 0.15$, at most 15% of the data points are not included in the prediction set, and at most 15% of the data can be regarded as the outliers. The default value for argument alpha.lim of hyperparam.alpha, which is M in Algorithm 3, is 0.15. We may interpret this level selecting procedure as finding the representative modes for the given mixture model; the chosen level is the cutoff value for which the most stable modes are not vanished.

In summary, we first choose the number of model components J in view of model selection, and then find the most stable level $\hat{\alpha}$ in the sense of invariability of the number of clusters. The function hyperparam.torus combines and implements Algorithms 2 and 3 sequentially and thus chooses J and α . This two-step hyperparameter selection procedure is used when mixture models are used to produce the conformal prediction sets, and can be invoked when the argument option of hyperparam.torus is set as option = "risk", "AIC", or "BIC". If option = "elbow" (the default value, if the dimension of data is $p = 2$), then the "elbow" criterion (12) is used to choose either (J, α) or (κ, α) . The function hyperparam.torus returns the chosen hyperparameters $(\hat{j}, \hat{\alpha})$ (or $(\hat{\kappa}, \hat{\alpha})$), as well as the corresponding model as an icp.torus object.

As an example, the following code applies the two-step procedure with option = "risk" to icp.torus.objects we evaluated earlier.

```
hyperparam.risk.out <- hyperparam.torus(icp.torus.objects, option = "risk")
hyperparam.risk.out

Type of conformity score: kmeans general
Optimizing method: risk
-----
Optimally chosen parameters. Number of components = 12 , alpha = 0.132716
Results based on criterion risk :
  J criterion
1  3 2016.575
2  4 1990.566
```

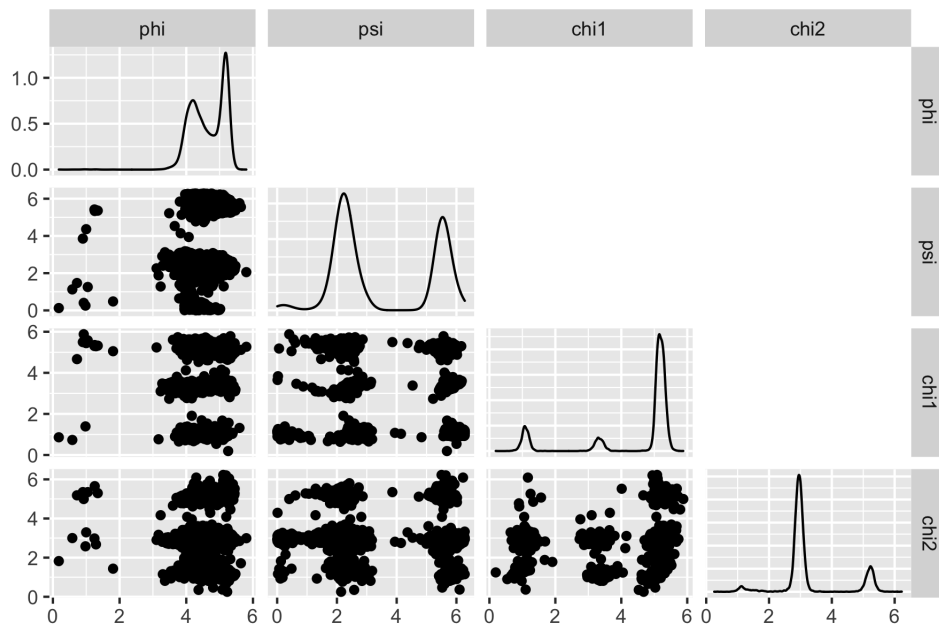


Figure 5: The pairwise scatter plots of ILE data, in which there are four variables (angles) ϕ , ψ , χ_1 and χ_2 . Each diagonal entry of the plot shows a marginal kernel density estimate for the corresponding angle. Each off-diagonal panel is a scatter plot for a pair of variables.

```
3  5 1907.887
4  6 1922.430
5  7 1924.768
... (omitted)
```

With the option "risk," $(\hat{j}, \hat{\alpha}) = (12, 0.01327)$. Recall that with option "elbow", we have chosen $(\hat{j}, \hat{\alpha}) = (12, 0.1111)$. The hyperparameter selection procedures can be visualized by `plot(hyperparam.out)` and `plot(hyperparam.risk.out)`. (The resulting graphic is omitted.)

In the next section, the two-step procedures for hyperparameter selection are used in a cluster analysis of data on \mathbb{T}^4 .

5 Clustering data on \mathbb{T}^4

In this section, we give an example of clustering ILE data in \mathbb{T}^4 . ILE is a dataset included in **ClusTorus**, which represents the structure of the isoleucine. This dataset is obtained by collecting several different '.pdb' files in the Protein Data Bank (Berman et al., 2003). We used PISCES (Wang and Dunbrack, 2003) to select high-quality protein data, by using several benchmarks—resolution is 1.6Å or better, R-factor is 0.22 or better, sequence percentage identity is equal to or less than 25—as described in Harder et al. (2010) and Mardia et al. (2012). The ILE data consist of $n = 8080$ instances of four angles $(\phi, \psi, \chi_1, \chi_2) \in \mathbb{T}^4$, and is displayed in Figure 5.

For predictive clustering of ILE data, the conformal prediction sets and scores are built from mixture models, fitted with the elliptical k -means algorithm (i.e., `model = "kmeans"`). Other choices of models such as "kde" and "mixture" are not applicable for this data set with $p > 2$. The number J of components in the mixture model needs to be tuned, and we set the candidates for J as $\{10, \dots, 40\}$. In the code example below, conformal prediction sets from mixture models are constructed by the function `icp.torus`, with `J = 10:40` indicating the candidates of J .

```
set.seed(2021)
icp.torus.objects <- icp.torus(ILE, J = 10:40)
```

Next step is to select the hyperparameter J , and the level α of the prediction set, using the function `hyperparam.torus`. As discussed in the previous section, for this data set with $p = 4$, evaluating the "elbow" criterion is computationally infeasible, and is not supported in `hyperparam.torus`, if $p > 2$. For $p > 2$, `hyperparam.torus` uses the two-step procedure, discussed in the previous section, with `option = "risk"` as the default choice for the criterion. In the code example below, we use the two-step procedure, but apply all three available criteria (`option = "risk", "AIC", and "BIC"`) in choosing \hat{j} .

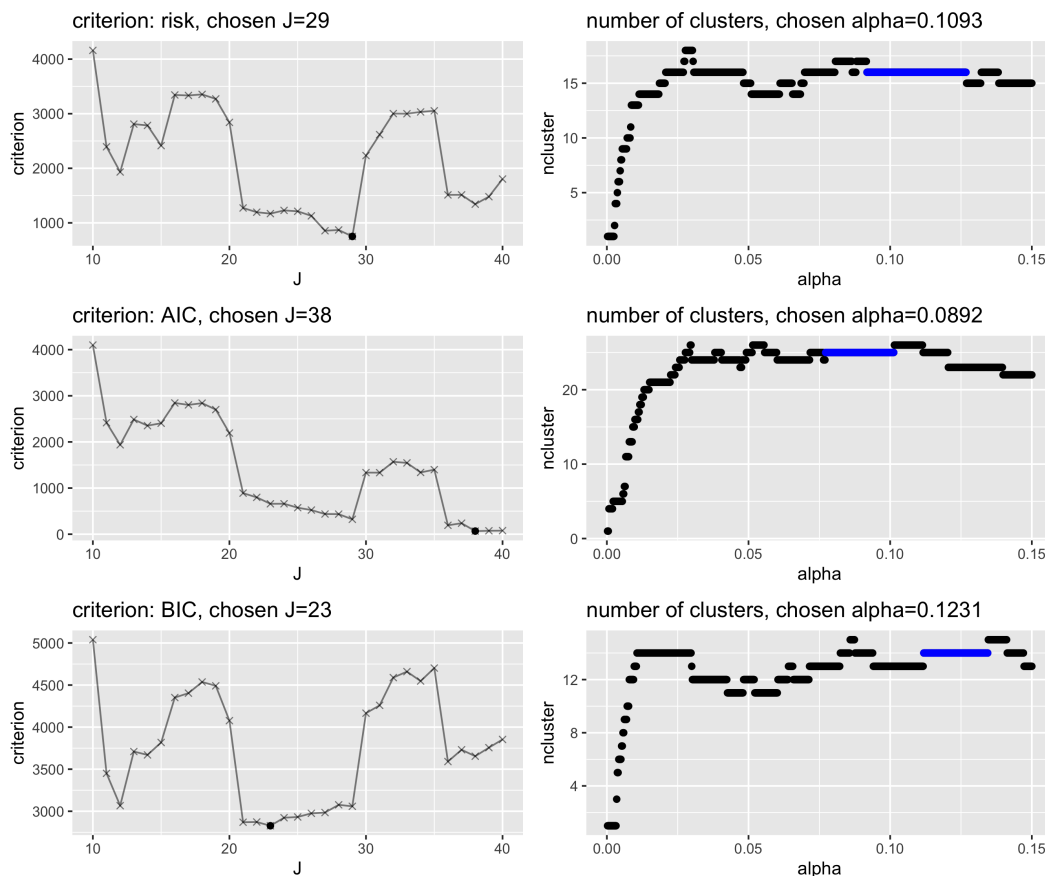


Figure 6: Hyperparameter selection for ILE data, generated from the outputs of hyperparam.torus. Rows correspond to different choices of criteria "risk", "AIC" and "BIC". In each row, the left panel shows the values of criterion over J , with the optimal \hat{J} indicated by a thicker dot; the right panel shows the number of clusters over varying α , in which the longest streak is highlighted. The optimal $\hat{\alpha}$ is the midpoint of the longest streak.

```
output_list <- sapply( c("risk", "AIC", "BIC"), function(opt) {
  hyperparam.torus(icp.torus.objects, option = opt)},
  simplify = FALSE,
  USE.NAMES = TRUE)
```

The result `output_list` is a list of length 3, consisting of outputs of the function `hyperparam.torus`. The details of hyperparameter selection can be visualized, and are shown in Figure 6. The first row of the figure is created by `plot(output_list$risk)`, and shows that the evaluated prediction risk is the smallest at $\hat{J} = 29$. On the right panel, it can be seen that the longest streak of the number of clusters over varying level α occurs at 16, which is given by a range of levels around $\hat{\alpha} = 0.1093$. The second and third rows are similarly generated, and they show the results of AIC- and BIC-based hyperparameter selection. While the results of hyperparameter selection from the three criteria do not always agree with each other, we observe that using BIC tends to choose parsimonious models than others, for this and many other data sets we tested.

The number of clusters, given by the conformal prediction set $C_n(\hat{\alpha}, \hat{J})$, can be seen in the right panels of Figure 6. For example, in the top right panel, with $\hat{J} = 29$ and $\hat{\alpha} = 0.1093$, the number of clusters is 16 (the vertical position of the blue-colored longest streak). For the subsequent analysis, we use the risk criterion, thus choosing $(\hat{J}, \hat{\alpha}) = (29, 0.1093)$.

```
hyperparam.risk.out <- output_list$risk
```

Finally, the function `cluster.assign.torus` is used for cluster membership assignment for each data point in ILE. In the code below, the function `cluster.assign.torus` takes as input `hyperparam.risk.out`, an output of `hyperparam.torus`, and we have not specified any level. Since the object `hyperparam.risk.out` contains the chosen level $\hat{\alpha}$ (in its value `alphahat`), the level of the conformal prediction set is, by default, set as `hyperparam.risk.out$alphahat`.

```
cluster.out <- cluster.assign.torus(hyperparam.risk.out)
```

The output `cluster.out` contains the membership assignment results as well as the number of clusters, which can be retrieved by `cluster.out$ncluster` or by simply printing the output `cluster.out`. The assigned cluster memberships can be displayed on the pairwise scatter plots of the four angles. We demonstrate the outlier-disposing membership assignment (the default behavior for S3 method `plot`), as well as the membership assignment based on the maximum of log-densities. Figure 7 displays the scatter plots generated by the codes:

```
plot(cluster.out, assignment = "outlier") # Top panel of Figure 7
plot(cluster.out, assignment = "log.density") # Bottom panel of Figure 7
```

Note that these cluster assignments are based on the conformal prediction set $C_n(\hat{\alpha}, \hat{f})$. The information to construct $C_n(\alpha, \hat{f})$ (for any $\alpha \in (0, 1)$) is contained in the object `hyperparam.risk.out` as value `icp.torus`. Since the conformal prediction set is a union of 4-dimensional toroidal ellipsoids, projections of such ellipsoids onto coordinate planes are plotted by the following code, and is shown in Figure 8.

```
set.seed(2021)
plot(hyperparam.risk.out$icp.torus,
     data = ILE[sample(1:nrow(ILE), 500), ],
     level = hyperparam.risk.out$alphahat)
```

Scatter plots of $n = 8080$ observations are typically too busy, especially when other information (such as the ellipses) is overlaid. In the code example above, we use the argument `data = ILE[sample(1:nrow(ILE), 500),]` to plot randomly selected observations.

6 All-in-one function: `clus.torus`

The predictive clustering for data on the torus is obtained by sequentially applying functions `icp.torus`, `hyperparam.torus` and `cluster.assign.torus`, as demonstrated for ILE data in the previous section. The function `clus.torus` is a user-friendly all-in-one function, which performs the predictive clustering by sequentially calling the three core functions.

Using `clus.torus` can be as simple as `clus.torus(data)`, as shown in the first code example, resulting in Figure 1, in Introduction. In this case, the three functions are called sequentially with default choices for their arguments. On the other hand, users can specify which models and fitting methods are used, whether hyperparameter tuning is required, and, if so, which criterion is used for `hyperparam.torus`, and so on. Key arguments of `clus.torus` are summarized in Table 3. The argument `model` only takes "kmeans" and "mixture" as input, which is passed to `icp.torus` inside the function. Since the function concerns clustering, conformal prediction sets consisting of ellipsoids (10) are needed, and such prediction sets are given by both `model = "kmeans"` and "mixture". Next, the values of the arguments `J` and `level` determine whether tuning is needed for hyperparameters J and α . If both are not specified, i.e., `J = NULL` and `level = NULL`, then `hyperparam.torus` is used to select both parameters, with argument `option` (see Table 3). If either `J` or `level` is specified as a scalar, then the function simply uses the given value for constructing the conformal prediction sets and for clustering. Other arguments available for `icp.torus` and `hyperparam.torus` can be specified, and the function passes those arguments to corresponding functions, if applicable.

The output of the function is a list of three objects, with S3 class `clus.torus`. The three objects in the output are

1. a `cluster.obj` object, containing the results of cluster membership assignments,
2. an `icp.torus` object, corresponding to the model with \hat{f} (or the specified `J`), and
3. if applicable, a `hyperparam.torus`, `hyperparam.J` or `hyperparam.alpha` object.

Each of these objects can be plotted via `plot`, defined for S3 class `clus.torus`. For example, recall that `ex` is a `clus.torus` object we created in Introduction. By setting the argument `panel` of the method `plot` as `panel = 1`, the `cluster.obj` object is plotted.

```
plot(ex, panel = 1) # equivalent to plot(ex)
```

The result is shown in Figure 1 (top left). If the data dimension is $p > 2$, then figures similar to Figure 7 will be created. If `panel = 2`, the `icp.torus` object is plotted, similar to Figures 3 and 8. Finally, if `panel = 3`, the graphics relevant to hyperparameter selection are created, similar to Figure 6.

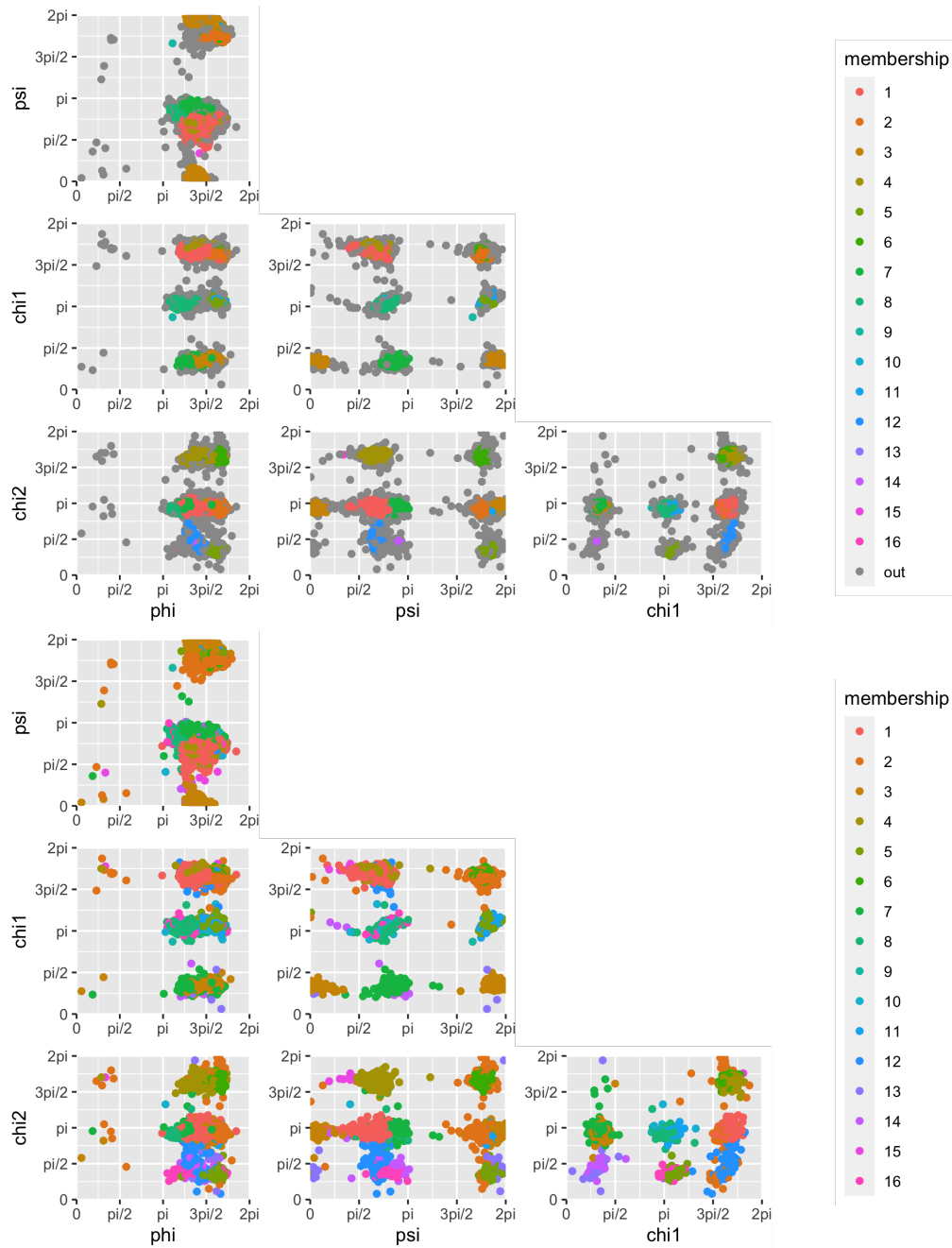


Figure 7: The pairwise scatter plots of ILE data with cluster assignments. (Top) assignment = "outlier". (Bottom) assignment = "log.density".

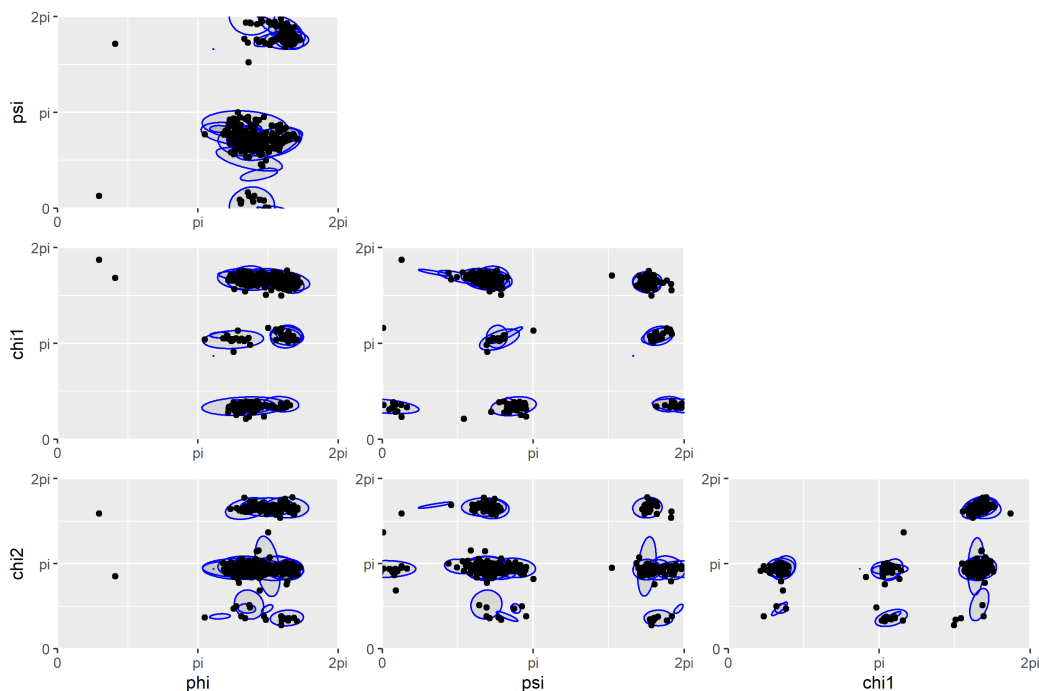


Figure 8: The pairwise scatter plots of ILE data, overlaid with the (projected) ellipsoids that constitute the conformal prediction set $C_n(\hat{\alpha}, \hat{f})$.

Arguments	Descriptions
data	$n \times d$ matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$
model	A string. One of "kmeans" and "mixture" which determines the model or estimation methods. If "mixture", the model is the von Mises mixture, fitted with an EM algorithm. If "kmeans", the model is also the von Mises mixture, fitted by the elliptical k-means algorithm. If the dimension of data space is greater than 2, only "kmeans" is supported. Default is model = "kmeans".
J	A scalar or numeric vector. If J is scalar, the number of components J is set as J. If J is a vector, then hyperparam. torus or hyperparam. J is used to select \hat{f} . Default is J = NULL, in which case J = 4:30 is used.
level	A scalar in $[0, 1]$. The level of the conformal prediction set used for clustering. Default is level = NULL, in which case hyperparam. alpha is used to choose optimal level $\hat{\alpha}$.
option	A string. One of "elbow", "risk", "AIC", or "BIC", determining the criterion used for hyperparam. torus and hyperparam. J. Default is option = "elbow" if $d = 2$, and option = "risk" if $d > 2$.

Table 3: Key arguments and descriptions of the function `clus.torus`

7 Other methods of clustering on the torus

Gao et al. (2018) and Jung et al. (2021) used the *extrinsic k-means*, which uses Euclidean embedding and enjoys fast computation of the vanilla *k-means* algorithm. That is, consider the mapping $f : \mathbb{T}^p \rightarrow \mathbb{R}^{2p}$ as

$$f(\phi_1, \dots, \phi_p) = (\cos \phi_1, \dots, \cos \phi_p, \sin \phi_1, \dots, \sin \phi_p)$$

which is the simple Euclidean embedding and is injective. Since \mathbb{R}^{2p} is a Euclidean space, the *k-means* clustering for vector-valued data can be used. The function `kmeans.torus` implements the extrinsic *k-means* clustering. In the simple code example below, the number of cluster is set to $k = 3$, and the result shows the membership assignment by the extrinsic *k-means* algorithm.

```
set.seed(2021)
exkmeans <- kmeans.torus(SARS_CoV_2, centers = 3, nstart = 30)
head(exkmeans$membership)

27.B.ALA  28.B.TYR  29.B.THR  30.B.ASN  31.B.SER  32.B.PHE
```

1 1 1 1 2 3

Distance-based clustering methods, such as hierarchical clustering, only requires a pairwise distances of the data points. The function `ang.pdist` generates the distance matrix for the input data in which the angular distance between the two points on \mathbb{T}^p is measured. Combined with `hclust`, the pairwise angular distances are used to provide a hierarchical clustering using, e.g., the complete linkage, as done in the following example.

```

dismat <- ang.pdist(SARS_CoV_2)
hc <- hclust(dismat, method = "complete")
hc.result <- cutree(hc, k = 3)
head(hc.result)

[1] 1 1 1 1 2 3
    
```

Figure 9 shows the results for the two clustering algorithms, discussed above. The left panel shows that the Euclidean embedding reflects the rotational nature of angular data. The right panel shows that the distance-based clustering methods is well-applied with `ang.pdist`. Note that both the extrinsic k -means and the hierarchical clustering results are invariant to different representations of angles. That is, the cluster assignments do not change if X_n is replaced by $X_n - \pi$. However, these methods are inadequate when true clusters are irregularly shaped and when there are outliers (Jung et al., 2021). In addition, the number of clusters needs to be predetermined for both methods. In contrast, these weaknesses are mostly resolved by using the predictive clustering.

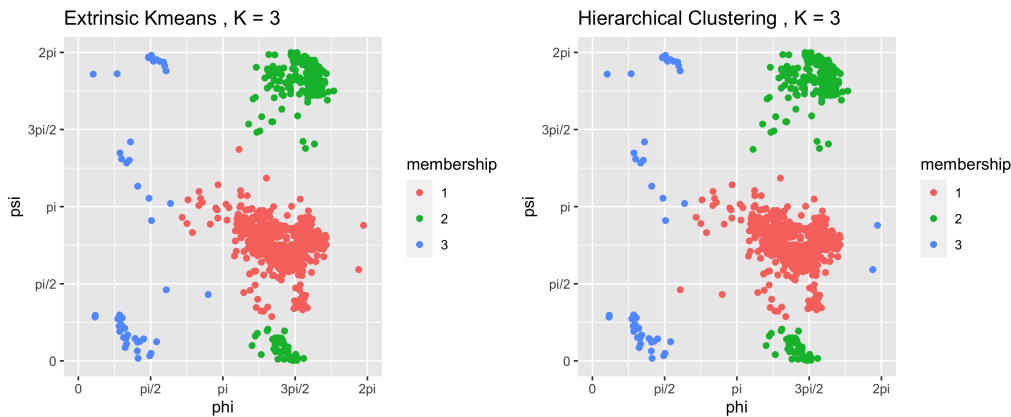


Figure 9: The clustering results for SARS-CoV-2 by using extrinsic k -means and hierarchical clustering under the 3 clusters assumption. The left panel shows the result for extrinsic k -means, and the right panel shows the result for hierarchical clustering.

8 Summary and discussion

In this paper, we introduced the package **ClusTorus** which contains various tools and routines for multivariate angular data, including kernel density estimates and mixture model estimates. **ClusTorus** performs clustering based on conformal prediction sets. We demonstrated our implementation with data on \mathbb{T}^4 . The clustering by **ClusTorus** can result in cluster assignment either with or without an outlier class. A reviewer pointed out that the package **MoEClust** (Murphy and Murphy, 2020, 2021) can also dispose some points as outliers. However, **MoEClust** only works on Euclidean space, not on \mathbb{T}^p .

There are some possible future developments for **ClusTorus**. First, EM algorithms for von Mises mixture models on high dimensional tori (e.g., \mathbb{T}^4) can be implemented assuming independence of angles in each component. Using closed-form approximations of maximum likelihood estimators for univariate von Mises-Fisher distributions (Banerjee et al., 2005; Hornik and Bettina, 2014), fitting mixtures of product components can be done efficiently (Grim, 2017). Another direction is obtained by viewing clustering based on (11) by varying α as surveying birth and death of connected components. This can be dealt with a persistence diagram, a concept of topological data analysis. Hence, instead of using Algorithm 3, one may choose desirable α using persistence diagram.

9 Appendix

Elliptical k -means algorithm

In this appendix, we outline the elliptical k -means algorithm for the data on the torus, implemented in the function `ellip.kmeans.torus`. The algorithm is used to estimate the parameters of the mixture model (5), approximated as in (8). Note that the EM algorithm can be used for parameter estimation for mixture models in low dimensions. The EM algorithms of Jung et al. (2021) is implemented in the function `EMsinvMmix`, but works for $p = 2$ only. For $p > 3$, EM algorithms suffer from high computational costs (Mardia et al., 2012). To circumvent this problem, we estimate the parameters by modifying the generalized Lloyd’s algorithm (Shin et al., 2019), also known as the elliptical k -means algorithm (Sung and Poggio, 1998; Bishop, 2006). For vector-valued data, Shin et al. (2019) showed that the elliptical k -means algorithm estimates the parameters sufficiently well for the max-mixture density case as (7).

Suppose $y_1, \dots, y_n \in \mathbb{T}^p$ are an independent and identically distributed sample. Using the approximated density (8), the approximated likelihood, L' , is

$$L'(\mu, \Sigma) = (2\pi)^{-np/2} |\Sigma|^{-n/2} \exp \left[-\frac{n}{2} \text{tr} \left(S \Sigma^{-1} \right) \right] \tag{13}$$

where $S = \frac{1}{n} \sum_{i=1}^n (y_i \ominus \mu) (y_i \ominus \mu)^T$. Thus, if μ is known, $\hat{\Sigma} = S$ maximizes L' . Following Mardia et al. (2012), the mean μ is estimated as follows. Let $\bar{U}_j = \sum_{i=1}^n \cos(y_{ij}) / n$ and $\bar{V}_j = \sum_{i=1}^n \sin(y_{ij}) / n$ for $j = 1, \dots, p$. Then, $\hat{\mu} = (\hat{\mu}_1, \dots, \hat{\mu}_p)^T$,

$$\hat{\mu}_j = \arctan \frac{\bar{V}_j}{\bar{U}_j}, \quad j = 1, \dots, p \tag{14}$$

which is the maximum likelihood estimator of mean direction of von Mises-Fisher distribution (Mardia and Jupp, 1999).

With these approximated maximum likelihood estimators, the elliptical k -means algorithm, described in Algorithm 4, maximizes the likelihood corresponding to the max-mixture model (7). The algorithm is implemented in the function `ellip.kmeans.torus`.

Algorithm 4 Elliptical k -means algorithm for the torus

- 1: **procedure** ELLIPTICAL K-MEANS($\{X_1, \dots, X_n\}, J$)
- 2: Initialize $\pi_j, \theta_j = (\mu_j, \Sigma_j), j = 1, \dots, J$
- 3: set

$$w_{i,j} = \begin{cases} 1, & \text{if } j = \arg \max_l \left[- (X_i \ominus \mu_l)^T \Sigma_l^{-1} (X_i \ominus \mu_l) - \log |\Sigma_l| + 2 \log \pi_l \right] \\ 0, & \text{otherwise} \end{cases}$$

$$I_j = \{i \in \{1, \dots, n\} \mid w_{i,j} = 1\}$$

- 4: Update μ_j as (14) with $\{X_i\}_{i \in I_j}$ for $j = 1, \dots, J$
 - 5: Update $\Sigma_j = \frac{1}{\sum_{i=1}^n w_{i,j}} \sum_{i=1}^n w_{i,j} (X_i \ominus \mu_j) (X_i \ominus \mu_j)^T$ for $j = 1, \dots, J$
 - 6: Update $\pi_j = \frac{1}{n} \sum_{i=1}^n w_{i,j}$ for $j = 1, \dots, J$
 - 7: Repeat step 3-6 until converge
 - 8: **end procedure**
-

Note that the initial values require an initial clustering. For this, we use other clustering algorithms such as the extrinsic k -means or the hierarchical clustering algorithms, and can be specified by argument `init` of `ellip.kmeans.torus` and `icp.torus`. One may specify arguments for either `hlcust` or `kmeans` in `icp.torus`. For example, one may specify the choice of initial values as follows.

```
icp.torus(data = SARS_CoV_2, J = 4, init = "kmeans", nstart = 30)
icp.torus(data = SARS_CoV_2, J = 4, init = "hierarchical", method = "complete")
```

By default, the hierarchical clustering with complete linkage is used. Data analysis in this article using `icp.torus` or `clus.torus` was performed with the default initialization.

Constraints for mixture models

The protein structure data we aim to analyze typically consist of hundreds of angles (observations). Fitting the mixture with a large number of components may give inefficient estimators. Thus, we have implemented options for reducing the number of model parameters, by constraining the shape of the ellipsoids, or the covariance matrices. Applying the constraints lead much faster convergence for estimating parameters (Grim, 2017). We list three types of constraints for covariance matrices Σ_j . These constraints are specified by setting the arguments `mixturefitmethod` and `kmeansfitmethod` (for `icp.torus`) and `type` (for `EMsinvMmix` and `ellip.kmeans.torus`). We explain in terms of the arguments for the function `icp.torus`.

- $\Sigma_j = \sigma_j^2 I_p$ for some $\sigma_j^2 > 0$ for all j , and the prediction set will be the union of spheres. `mixturefitmethod = "circular"` and `kmeansfitmethod = "heterogeneous-circular"` represents this constraint. Furthermore, if $\sigma_1^2 = \dots = \sigma_J^2$ and $\pi_j = 1/J$ for all j , then all the spheres have the same radii and this constraint can be designated with `kmeansfitmethod = "homogeneous-circular"`.
- $\Sigma_j = \text{diag}(\sigma_{jk}^2)_{k=1, \dots, p}$ for $\sigma_{jk}^2 > 0$, and the fitted ellipsoids \hat{E}_j ($j = 1, \dots, J$) are the axis-aligned ellipsoids. `mixturefitmethod = "axis-aligned"` represents this constraint.
- No constraint for Σ_j , and \hat{E}_j ($j = 1, \dots, J$) are any ellipsoids. This option can be designated by `mixturefitmethod = "general"` and `kmeansfitmethod = "general"`.

The default values for `icp.torus` are `kmeansfitmethod = "general"` and `mixturefitmethod = "axis-aligned"`.

List of S3 classes defined in ClusTorus

Several S3 classes are defined in the packages **ClusTorus**. A list of the S3 classes is given in Table 4.

S3 class	functions	methods
<code>cp.torus.kde</code>	<code>cp.torus.kde</code>	<code>print, plot</code>
<code>icp.torus</code>	<code>icp.torus</code>	<code>print, plot, LogLik, predict</code>
<code>icp.torus.eval</code>	<code>icp.torus.eval</code>	<code>print</code>
<code>cluster.obj</code>	<code>cluster.assign.torus</code>	<code>print, plot</code>
<code>kmeans.torus</code>	<code>kmeans.torus</code>	<code>print, predict</code>
<code>hyperparam.torus</code>	<code>hyperparam.torus</code>	<code>print, plot</code>
<code>hyperparam.J</code>	<code>hyperparam.J</code>	<code>print, plot</code>
<code>hyperparam.alpha</code>	<code>hyperparam.alpha</code>	<code>print, plot</code>
<code>clus.torus</code>	<code>clus.torus</code>	<code>print, plot</code>

Table 4: List of S3 classes, the functions returning a list with corresponding S3 class, and available methods for the S3 class.

10 Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C2002256).

Bibliography

- H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. URL <https://doi.org/10.1109/TAC.1974.1100705>. [p196]
- A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(46):1345–1382, 2005. URL <http://jmlr.org/papers/v6/banerjee05a.html>. [p203]
- T. Benaglia, D. Chauveau, D. R. Hunter, and D. Young. `mixtools`: An R package for analyzing finite mixture models. *Journal of Statistical Software*, 32(6):1–29, 2009. URL <http://www.jstatsoft.org/v32/i06/>. [p187]

- H. Berman, K. Henrick, and H. Nakamura. Announcing the worldwide protein data bank. *Nature Structural and Molecular Biology*, 10:980, 2003. URL <https://doi.org/10.1038/nsb1203-980>. [p186, 198]
- C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, Inc., 33 Spring Street, New York, NY 10013, USA, 2006. [p186, 193, 204]
- C. Branden and J. Tooze. *Introduction to Protein Structure*. Garland Publishing, Inc, 19 Union Square West, New York, 1999. [p186]
- S. Chakraborty and S. W. Wong. *BAMBI: Bivariate Angular Mixture Models*, 2020. URL <https://CRAN.R-project.org/package=BAMBI>. R package version 2.3.0. [p186]
- S. Chakraborty and S. W. K. Wong. BAMBI: An R package for Fitting Bivariate Angular Mixture Models, 2019. URL <https://arxiv.org/abs/1708.07804>. [p186]
- Coronaviridae Study Group of the International Committee on Taxonomy of Viruses., A. e. Gorbalenya, S. C. baker, R. S. baric, R. J. de Groot, C. Drosten, A. A. Gulyaeva, bart I. Haagmans, C. lauber, A. M. leontovich, benjamin W. Neuman, D. Penzar, S. Perlman, leo I. M. Poon11, D. V. Samborskiy, I. A. Sidorov, I. Sola, and J. Ziebuhr. The species severe acute respiratory syndrome- related coronavirus: classifying 2019-ncov and naming it sars-cov-2. *Nature Microbiology*, 5:536—544, 2020. URL <https://doi.org/10.1038/s41564-020-0695-z>. [p186]
- Y. Gao, S. Wang, and M. Deng. Raptorx-angle: real-value prediction of protein backbone dihedral angles through a hybrid method of clustering and deep learning. *BMC Biometrics*, 19(100), 2018. URL <https://doi.org/10.1186/s12859-018-2065-x>. [p187, 202]
- I. Gilitschenski and U. D. Hanebeck. A robust computational test for overlap of two arbitrary-dimensional ellipsoids in fault-detection of kalman filters. In *2012 15th International Conference on Information Fusion*, pages 396–401, 2012. URL <https://ieeexplore.ieee.org/document/6289830>. [p194]
- B. Grant, X.-Q. Yao, L. Skjaerven, and J. Ide. *bio3d: Biological Structure Analysis*, 2021. URL <https://CRAN.R-project.org/package=bio3d>. R package version 2.4-2. [p186]
- B. J. Grant, A. P. C. Rodrigues, K. M. ElSawy, J. A. McCammon, and L. S. D. Caves. Bio3d: an R package for the comparative analysis of protein structures. *Bioinformatics*, 22(21):2695–2696, 08 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl461. URL <https://doi.org/10.1093/bioinformatics/btl461>. [p186]
- J. Grim. Approximation of unknown multivariate probability distributions by using mixtures of product components: A tutorial. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(09):1750028, 2017. [p203, 205]
- T. Harder, W. Boomsma, M. Paluszewski, J. Frelsen, K. E. Johansson, and T. Hamelryck. Beyond rotamers: a generative, probabilistic model of side chains in proteins. *BMC Bioinformatics*, 11:306, 2010. URL <https://doi.org/10.1186/1471-2105-11-306>. [p198]
- K. Hornik and G. Bettina. On maximum likelihood estimation of the concentration parameter of von mises–fisher distributions. *Computational Statistics*, 29:945—957, 2014. URL <https://doi.org/10.1007/s00180-013-0471-0>. [p203]
- S. Jung and S. Hong. *ClusTorus: Prediction and Clustering on the Torus by Conformal Prediction*, 2021. URL <https://CRAN.R-project.org/package=ClusTorus>. R package version 0.2.1. [p186]
- S. Jung, K. Park, and B. Kim. Clustering on the torus by conformal prediction. *Annals of Applied Statistics*, 15(4):1583–1603, 2021. URL <https://doi.org/10.1214/21-AOAS1459>. [p187, 188, 190, 191, 193, 194, 195, 202, 203, 204]
- J. Lei, J. Robins, and L. Wasserman. Distribution-free prediction sets. *Journal of the American Statistical Association*, 108(501):278–287, 2013. doi: 10.1080/01621459.2012.751873. URL <https://doi.org/10.1080/01621459.2012.751873>. [p188, 195, 196]
- J. Lei, A. Rinaldo, and L. Wasserman. A conformal prediction approach to explore functional data. *Ann Math Artif Intell*, 74:29–43, 2015. URL <https://doi.org/10.1007/s10472-013-9366-63>. [p188, 189]
- S. C. Lovell, I. W. Davis, W. B. Arendall III, P. I. De Bakker, J. M. Word, M. G. Prisant, J. S. Richardson, and D. C. Richardson. Structure validation by α geometry: ϕ , ψ and $c\beta$ deviation. *Proteins: Structure, Function, and Bioinformatics*, 50(3):437–450, 2003. [p186]

- K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley, New York, 1999. [p204]
- K. V. Mardia, C. C. Taylor, and G. K. Subramaniam. Protein bioinformatics and mixtures of bivariate von mises distributions for angular data. *Biometrics*, 63(2):505—512, 2007. URL <https://doi.org/10.1111/j.1541-0420.2006.00682.x>. [p186]
- K. V. Mardia, G. Hughes, C. C. Taylor, and H. Singh. A multivariate von mises distribution with applications to bioinformatics. *The Canadian Journal of Statistics / La Revue Canadienne de Statistique*, 36(1):99–109, 2008. URL <http://www.jstor.org/stable/20445295>. [p190]
- K. V. Mardia, J. T. Kent, Z. Zhang, and C. C. T. . T. Hamelryck. Mixtures of concentrated multivariate sine distributions with applications to bioinformatics. *Journal of Applied Statistics*, 39(11):2475–2492, 2012. URL <https://doi.org/10.1080/02664763.2012.719221>. [p186, 187, 191, 198, 204]
- M. D. Marzio, A. Panzera, and C. C. Taylor. Kernel density estimation on the torus. *Journal of Statistical Planning and Inference*, 141(6):2156–2173, 2011. ISSN 0378-3758. doi: <https://doi.org/10.1016/j.jspi.2011.01.002>. URL <https://www.sciencedirect.com/science/article/pii/S037837581100019X>. [p190]
- K. Murphy and T. B. Murphy. Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 14(2):293–325, 2020. URL <https://doi.org/10.1007/s11634-019-00373-8>. [p203]
- K. Murphy and T. B. Murphy. *MoEClust: Gaussian Parsimonious Clustering Models with Covariates and a Noise Component*, 2021. URL <https://cran.r-project.org/package=MoEClust>. R package version 1.4.2. [p203]
- K. Oberholser. Proteopedia entry: Ramachandran plots. *Biochemistry and Molecular Biology Education*, 38(6):430–430, 2010. [p186]
- G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461 – 464, 1978. URL <https://doi.org/10.1214/aos/1176344136>. [p196]
- L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):289–317, 2016. URL <https://doi.org/10.32614/RJ-2016-021>. [p187]
- J. Shin, A. Rinaldo, and L. Wasserman. Predictive clustering, 2019. URL <https://arxiv.org/abs/1903.08125>. [p186, 187, 188, 191, 193, 196, 204]
- K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998. URL <https://doi.org/10.1109/34.655648>. [p186, 193, 204]
- V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. Springer Science+Business Media, Inc., 33 Spring Street, New York, NY 10013, USA, 2005. [p186, 187, 188, 189]
- A. C. Walls, Y.-J. Park, M. A. Tortorici, A. Wall, A. T. McGuire, and D. Veasley. Structure, function, and antigenicity of the sars- cov-2 spike glycoprotein. *Cell*, 181, 2020. URL <https://doi.org/10.1016/j.cell.2020.02.058>. [p187]
- G. Wang and J. Dunbrack, Roland L. PISCES: a protein sequence culling server. *Bioinformatics*, 19(12):1589–1591, 08 2003. ISSN 1367-4803. URL <https://doi.org/10.1093/bioinformatics/btg224>. [p198]

Seungki Hong
Department of Statistics, Seoul National University
1, Gwanak-ro, Gwanak-gu, Seoul
Republic of Korea
skgaboja@snu.ac.kr

Sungkyu Jung
Department of Statistics, Seoul National University
1, Gwanak-ro, Gwanak-gu, Seoul
Republic of Korea
sungkyu@snu.ac.kr

kStatistics: Unbiased Estimates of Joint Cumulant Products from the Multivariate Faà Di Bruno's Formula

by *Elvira Di Nardo and Giuseppe Guarino*

Abstract `kStatistics` is a package in R that serves as a unified framework for estimating univariate and multivariate cumulants as well as products of univariate and multivariate cumulants of a random sample, using unbiased estimators with minimum variance. The main computational machinery of `kStatistics` is an algorithm for computing multi-index partitions. The same algorithm underlies the general-purpose multivariate Faà di Bruno's formula, which therefore has been included in the last release of the package. This formula gives the coefficients of formal power series compositions as well as the partial derivatives of multivariable function compositions. One of the most significant applications of this formula is the possibility to generate many well-known polynomial families as special cases. So, in the package, there are special functions for generating very popular polynomial families, such as the Bell polynomials. However, further families can be obtained, for suitable choices of the formal power series involved in the composition or when suitable symbolic strategies are employed. In both cases, we give examples on how to modify the R codes of the package to accomplish this task. Future developments are addressed at the end of the paper

1 Introduction

Joint cumulants are usually employed for measuring interactions among two or more random variables simultaneously, extending the familiar notion of covariance to higher orders. More in details, suppose \mathbf{Y} a random vector with moment generating function $M_{\mathbf{Y}}(\mathbf{z})$, for $\mathbf{z} = (z_1, \dots, z_m)$ in a suitable neighborhood of $\mathbf{0}$. Thus $M_{\mathbf{Y}}(\mathbf{z})$ can be expressed as

$$M_{\mathbf{Y}}(\mathbf{z}) = \exp(K_{\mathbf{Y}}(\mathbf{z})) \quad (1)$$

where $K_{\mathbf{Y}}(\mathbf{z})$ is the cumulant generating function of \mathbf{Y} . If $\mathbf{i} \in \mathbb{N}_0^m$ and

$$M_{\mathbf{Y}}(\mathbf{z}) = 1 + \sum_{|\mathbf{i}|>0} \frac{\mathbb{E}[\mathbf{Y}^{\mathbf{i}}]}{\mathbf{i}!} \mathbf{z}^{\mathbf{i}} \quad K_{\mathbf{Y}}(\mathbf{z}) = \sum_{|\mathbf{i}|>0} \frac{k_{\mathbf{i}}(\mathbf{Y})}{\mathbf{i}!} \mathbf{z}^{\mathbf{i}} \quad (2)$$

then $\{k_{\mathbf{i}}(\mathbf{Y})\}$ are said the joint cumulants of $\{\mathbb{E}[\mathbf{Y}^{\mathbf{i}}]\}$. From a theoretical point of view, cumulants are a useful sequence due to the following properties (Elvira Di Nardo 2011):

- *Orthogonality*: Joint cumulants of independent random vectors are zero, that is $k_{\mathbf{i}}(\mathbf{Y}) = 0$ for $|\mathbf{i}| > 0$ if $\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2)$ with \mathbf{Y}_1 independent of \mathbf{Y}_2 .
- *Additivity*: Cumulants linearize on independent random vectors, that is $k_{\mathbf{i}}(\mathbf{Y}_1 + \mathbf{Y}_2) = k_{\mathbf{i}}(\mathbf{Y}_1) + k_{\mathbf{i}}(\mathbf{Y}_2)$ for $|\mathbf{i}| > 0$ with \mathbf{Y}_1 independent of \mathbf{Y}_2 .
- *Multilinearity*: $k_{\mathbf{i}}(A\mathbf{Y}) = \sum_{j_1, \dots, j_m} (A)_{i_1}^{j_1} \cdots (A)_{i_m}^{j_m} k_{\mathbf{j}}(\mathbf{Y})$ for $|\mathbf{i}| > 0$ with $A \in \mathbb{R}^m \times \mathbb{R}^m$.
- *Semi-invariance*: If $\mathbf{b} \in \mathbb{R}^m$ then $k_{\mathbf{i}}(\mathbf{Y} + \mathbf{b}) = k_{\mathbf{i}}(\mathbf{Y})$ for $|\mathbf{i}| \geq 2$.

Thanks to all these properties, joint cumulants have a wide range of applications: from statistical inference and time series (Jammalamadaka, Rao, and Terdik 2006) to asymptotic theory (Rao and Wong 1999), from spatial statistics modeling (Dimitrakopoulos, Mustapha, and Gloaguen 2010) to signal processing (Giannakis 1987), from non-linear systems identification (Oualla et al. 2021) to Wiener chaos (Peccati and Taqqu 2011), just to mention a few. Indeed it is also well known that cumulants of order greater than two are zero for random vectors which are Gaussian. Therefore, higher order cumulants are often used in testing for multivariate Gaussianity (Jammalamadaka, Rao, and Terdik 2006).

The i -th multivariate k -statistic is a symmetric function of the multivariate random sample whose expectation is the joint cumulant of order i of the population characters. These estimators have minimum variance when compared to all other unbiased estimators and are built by free-distribution methods without using sample moments. Due to the properties of joint cumulants, multivariate k -statistics are employed to check multivariate gaussianity (Ferreira, Magueijo, and Silk 1997) or

¹If $\mathbf{i} \in \mathbb{N}_0^m$ is a multi-index then we set $\mathbf{i}! = i_1! \cdots i_m!$ and $|\mathbf{i}| = i_1 + \cdots + i_m$.

to quantify high-order interactions among data (Geng, Liang, and Wang 2011), for applications in topology inference (Smith et al. 2022), in neuronal science (Staude, Rotter, and Grün 2010) and in mathematical finance (E. Di Nardo, Marena, and Semeraro 2020). Polykays are unbiased estimators of cumulant products (Robson 1957) and are particularly useful in estimating covariances between k -statistics (McCullagh 1987). In the **kStatistics** package (E. Di Nardo and Guarino 2021), the `nPolyk` function provides k -statistics and polykays as well as their multivariate generalizations. Further implementations are in `Phyton` (Smith 2020), in `Maple` (Guarino, Senato, and Di Nardo 2009) and in `Mathematica` (Rose and Smith 2002).

All these estimators are described with a wealth of details by Stuart and Ord (1994) and McCullagh (1987) and their construction relied on some well-known change of bases in the ring of symmetric polynomials. In Elvira Di Nardo (2011) a different approach is followed using suitable polynomial families and symbolic strategies. This procedure was the core of the first release (version 1.0) of the **kStatistics** package (E. Di Nardo and Guarino 2019), as the initial goal was to implement tools for the estimation of cumulants and cumulant products, both in the univariate and in the multivariate case. As the referred polynomial families can be traced back to the generalized (complete exponential) Bell polynomials, the latest version of the package (E. Di Nardo and Guarino 2021) has also included procedures to generate these polynomials together with a number of special cases.

Let us recall that the generalized (complete exponential) Bell polynomials are a family of polynomials involving multivariable Sheffer sequences (Brown 1979). Among its various applications, we recall the cumulant polynomial sequences and their connection with special families of stochastic processes (E. Di Nardo 2016a). Indeed, cumulant polynomials allow us to compute moments and cumulants of multivariate Lévy processes (E. Di Nardo and Oliva 2011), subordinated multivariate Lévy processes (E. Di Nardo, Marena, and Semeraro 2020) and multivariate compound Poisson processes (E. Di Nardo 2016b). Further examples can be found in Reiner (1976), Shrivastava (2002), Withers and Nadarajah (2010) or Privault (2021).

The generalized (complete exponential) Bell polynomials arise from the multivariate Faà di Bruno's formula, whose computation has been included in the latest version of the **kStatistics** package. In enumerative combinatorics, Faà di Bruno's formula is employed in dealing with formal power series. In particular the multivariate Faà di Bruno's formula gives the i -th coefficient of the composition (E. Di Nardo, Guarino, and Senato 2011)

$$h(z) = f(g_1(z) - 1, \dots, g_n(z) - 1) \tag{3}$$

where f and g_j for $j = 1, \dots, n$ are (exponential) formal power series

$$f(x) = \sum_{|t| \geq 0} f_t \frac{x^t}{t!} \quad \text{and} \quad g_j(z) = \sum_{|s| \geq 0} g_{j;s} \frac{z^s}{s!}, \tag{4}$$

with $x = (x_1, \dots, x_n)$, $z = (z_1, \dots, z_m)$ and $x^t = x_1^{t_1} \dots x_n^{t_n}$, $z^s = z_1^{s_1} \dots z_m^{s_m}$, $f_t = f_{t_1, \dots, t_n}$, $g_{j;s} = g_{j;s_1, \dots, s_m}$ for $j = 1, \dots, n$, and $f_0 = g_{1;0} = \dots = g_{n;0} = 1$. For instance, from (1) and (2) joint moments can be recovered from joint cumulants using the multivariate Faà di Bruno's formula for $n = 1$, $g(z) = 1 + K_Y(z)$ and $f(x) = \exp(x)$. As $1 + K_Y(z) = 1 + \log([M_Y(z) - 1] + 1)$ then joint cumulants can be recovered from joint moments using the multivariate Faà di Bruno's formula for $n = 1$, $g(z) = M_Y(z)$ and $f(x) = 1 + \log(1 + x)$. Let us remark that the exponential form (4) of the formal power series f and $\{g_j\}$ is not a constraint. To work with ordinary formal power series, the multi-index sequence $\{f_t\}$ needs to be replaced by the sequence $\{t!f_t\}$ as well as the multi-index sequence $\{g_{j;s}\}$ by the sequence $\{s!g_{j;s}\}$ for $j = 1, \dots, n$. In this case, the multivariate Faà di Bruno's formula gives the coefficient $i!\tilde{h}_i$ with \tilde{h}_i the i -th coefficient of the (ordinary) formal power series composition (3).

The problem of finding suitable and easily manageable expressions of the multivariate Faà di Bruno's formula has received attention from several researchers over the years. This is because the multivariate Faà di Bruno's formula is a very general-purpose tool with many applications. We refer to the paper of Leipnik and Pearce (2007) for a detailed list of references on this subject and a detailed account of its applications. Further applications can be found in Savits (2006), Chacón and Duong (2015), Shabat and Efendiev (2017) and Nguwui, Penent, and Privault (2022). A classical way to generate the multivariate Faà di Bruno's formula involves the partial derivatives of a composition of multivariable functions. Suppose $f(x)$ and $g_1(z), \dots, g_n(z)$ in (3) be differentiable functions a certain number of times. The multivariate Faà di Bruno's formula gives the partial derivative of order i of $h(z)$ in z_0

$$h_i = \frac{\partial^{|i|}}{\partial z_1^{i_1} \dots \partial z_m^{i_m}} h(z_1, \dots, z_m) \Big|_{z=z_0} \quad \text{for } |i| > 0, \tag{5}$$

²We use these notations independently if the powers or the subscripts are row vectors or column vectors.

assuming the partial derivatives of order \mathbf{t} of $f(x)$ exist in $x_0 = (g_1(z_0), \dots, g_n(z_0))$

$$f_{\mathbf{t}} = \frac{\partial^{|\mathbf{t}|}}{\partial x_1^{t_1} \dots \partial x_n^{t_n}} f(x_1, \dots, x_n) \Big|_{x=x_0} \quad \text{for } 0 < |\mathbf{t}| \leq |\mathbf{i}|,$$

and the partial derivatives of order \mathbf{s} of $g_j(z)$ exist in z_0 for $j = 1, \dots, n$

$$g_{j,\mathbf{s}} = \frac{\partial^{|\mathbf{s}|}}{\partial z_1^{s_1} \dots \partial z_m^{s_m}} g_j(z_1, \dots, z_m) \Big|_{z=z_0} \quad \text{for } 0 < |\mathbf{s}| \leq |\mathbf{i}|.$$

There are various ways to express h_i in (5), see for example Mishkov (2000), Hernández Encinas and Muñoz Masqué (2003) and Ma (2009). Symbolic manipulation using *MacSyma*, *Maple*, *Mathematica*, etc. can produce any required order of (5), by applying the chain rule recursively and using a function that provides partial derivatives. Also in *R*, there are some functions for computing partial derivatives (Clausen and Sokol 2020). Despite its conceptual simplicity, applications of the chain rule become impractical for its cumbersome computation even for small values of its order. As the number of additive terms becomes huge, the output is often untidy and further manipulations are required to simplify the result. By using combinatorial methods, Constantine and Savits (1996) have carried out the following expression of the multivariate Faà di Bruno’s formula

$$h_i = \mathbf{i}! \sum_{1 \leq |\mathbf{t}| \leq |\mathbf{i}|} f_{\mathbf{t}} \sum_{k=1}^{|\mathbf{i}|} \sum_{p_k(\mathbf{i}, \mathbf{t})} \prod_{j=1}^k \frac{(g_{l_j})^{q_j}}{q_j! (l_j!)^{|q_j|}} \tag{6}$$

where $(g_{\mathbf{s}})^q = \prod_{j=1}^m (g_{j,s})^{q_j}$ with $\mathbf{q} = (q_1, \dots, q_n)$ and

$$p_k(\mathbf{i}, \mathbf{t}) = \left\{ (q_1, \dots, q_k; l_1, \dots, l_k) : |q_j| > 0, \sum_{j=1}^k q_j = \mathbf{t}, \sum_{j=1}^k |q_j| l_j = \mathbf{i} \right\}$$

with $q_1, \dots, q_k \in \mathbb{N}_0^n$ and $l_1, \dots, l_k \in \mathbb{N}_0^m$ such that³ $\mathbf{0} \prec l_1 \prec \dots \prec l_k$.

A completely different approach concerns the combinatorics of partial derivatives as Hardy (2006) pointed out for the univariate-multivariate composition using multisets and collapsing partitions. Motivated by his results and using the umbral calculus, which is a symbolic method particularly useful in dealing with formal power series (4), the combinatorics behind (6) has been simplified and a different expression has been given in E. Di Nardo, Guarino, and Senato (2011). The key tool is the notion of partition of a multi-index which parallels the multiset partitions given in Hardy (2006).

The contribution of this paper is multi-sided. We explain how to recover in *R* a multi-index partition, which is a combinatorial device. For statistical purposes, we show how to recover k -statistics and their multivariate generalizations using the referred polynomial approach and multi-index partitions. Then, we explain the main steps of the MFB function producing the multivariate Faà di Bruno’s formula, without any reference to the umbral calculus or chain rules and whose applications go beyond statistical purposes. The main idea is to expand the multivariable polynomial

$$\sum \binom{\mathbf{i}}{s_1, \dots, s_n} q_{1,s_1}(y_1) \dots q_{n,s_n}(y_n)$$

where $q_{1,s_1}(y_1) \dots q_{n,s_n}(y_n)$ are suitable polynomials and the sum is over all the compositions of \mathbf{i} in n parts, that is all the n -tuples (s_1, \dots, s_n) of non-negative integer m -tuples such that $s_1 + \dots + s_n = \mathbf{i}$. Readers interested in the umbral setting may refer to Elvira Di Nardo (2011) and references therein.

Consequently, the MFB function gives an efficient computation of the following compositions:

- univariate with univariate, that is $n = m = 1$;
- univariate with multivariate, that is $n = 1$ and $m > 1$;
- multivariate with univariate, that is $n > 1$ and $m = 1$;
- multivariate with multivariate, that is $n > 1$ and $m > 1$.

The **kStatistics** package includes additional functions, for some of the most widespread applications of the multivariate Faà di Bruno’s formula. Indeed, not only this formula permits to generate joint cumulants and their inverse relations, but also further general families of polynomials. Therefore, we have set up special procedures for those families used very often in applications. These functions

³If $\mu, \nu \in \mathbb{N}_0^m$ we have $\mu \prec \nu$ if $|\mu| < |\nu|$ or $|\mu| = |\nu|$ and $\mu_1 < \nu_1$ or $|\mu| = |\nu|$ and $\mu_1 = \nu_1, \dots, \mu_k = \nu_k, \mu_{k+1} < \nu_{k+1}$ for some $1 \leq k < m$.

should be considered an easy to manage interfaces of the MFB function, with the aim of simplifying its application. Moreover, since the R codes are free, the user might follow similar steps to generate polynomial families not included in the package but always coming from the multivariate Faà di Bruno’s formula. The construction of new families of polynomials can be done mainly in two ways. The first way is to choose appropriately the coefficients $\{f_t\}$ and $\{g_{j;s}\}$ in (4). The second way is to use some suitable symbolic strategies, as discussed in Elvira Di Nardo (2011). For both cases, we provide examples.

The paper is organized as follows. The next section explains the main steps of the algorithm that produces multi-index partitions with particular emphasis on its combinatorics. Then we present the symbolic strategy to generate k -statistics and their generalizations using suitable polynomial sequences and multi-index partitions. The subsequent section deals with generalized (complete exponential) Bell polynomials and some special cases corresponding to well-known families of polynomials. We have also included the procedures to generate joint cumulants from joint moments and vice versa. In the last section we explain the main steps of the algorithm to produce the multivariate Faà di Bruno’s formula. We give examples of how to build new polynomials not included in the package. Some concluding remarks end the paper.

2 Partitions of a multi-index

Most routines of the `kStatistics` package use the partitions of a multi-index i . Therefore, before describing any of these routines, we recall the notion of multi-index partition and describe the algorithm for its construction as implemented in the `mkmSet` function of the package.

A partition of the multi-index $i = (i_1, \dots, i_m) \in \mathbb{N}_0^m$ is a matrix $\Lambda = (\lambda_1^{r_1}, \lambda_2^{r_2}, \dots)$ of non-negative integers with m rows and no zero columns such that

- $r_1 \geq 1$ columns are equal to λ_1 , $r_2 \geq 1$ columns are equal to λ_2 and so on;
- the columns $\lambda_1 < \lambda_2 < \dots$ are in lexicographic order⁴;
- the sum of the integers in the t -th row is equal to i_t , that is $\lambda_{t1} + \lambda_{t2} + \dots = i_t$ for $t = 1, 2, \dots, m$.

We write $\Lambda \vdash i$ to denote that Λ is a partition of i . Some further notations are:

- $\mathbf{m}(\Lambda) = (r_1, r_2, \dots)$, the vector of multiplicities of $\lambda_1, \lambda_2, \dots$
- $l(\Lambda) = |\mathbf{m}(\Lambda)| = r_1 + r_2 + \dots$, the number of columns of Λ with $l(\Lambda) = 0$ if $\Lambda \vdash \mathbf{0}$
- $\Lambda! = (\lambda_1!)^{r_1} (\lambda_2!)^{r_2} \dots$

Example 1: The partitions of $i = (2, 1)$ are the matrices

$$\begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} = (\lambda_1, \lambda_2^2),$$

with

$$\lambda_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{and} \quad \lambda_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The algorithm to get all the partitions of a multi-index resorts to multiset subdivisions. Let’s start by recalling the notion of multiset. A multiset M is a “set with multiplicities”. Suppose $a \in M$. Then the multiplicity of a is the number of times a occurs in M as a member. For example, the integers 3 and 2 are the multiplicities of a and b respectively in $M = \{a, a, a, b, b\}$. A subdivision of the multiset M is a multiset of sub-multisets of M , such that their disjoint union returns M . Examples of subdivisions of $M = \{a, a, a, b, b\}$ are

$$S_1 = \{\{a\}, \{a, b\}, \{a, b\}\}, \quad S_2 = \{\{a\}, \{a, a, b\}, \{b\}\}, \tag{7}$$

$$S_3 = \{\{a\}, \{a, a\}, \{b\}, \{b\}\}. \tag{8}$$

The subdivisions of the multiset $M = \{a, a, a, b, b\}$ are in one-to-one correspondence with the partitions $\Lambda \vdash (3, 2)$. For example, the subdivisions (7) correspond to the partitions $\Lambda_1 = (\lambda_2, \lambda_3^2)$ and $\Lambda_2 = (\lambda_1, \lambda_2, \lambda_5)$ respectively, while (8) to $\Lambda_3 = (\lambda_1^2, \lambda_2, \lambda_4)$ with

$$\lambda_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \{b\} \quad \lambda_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \{a\}$$

⁴As example $(a_1, b_1) < (a_2, b_2)$ if $a_1 < a_2$ or $a_1 = a_2$ and $b_1 < b_2$.

$$\lambda_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \{a, b\} \quad \lambda_4 = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \rightarrow \{a, a\} \quad \lambda_5 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \rightarrow \{a, a, b\}.$$

Multiset subdivisions can be recovered by using collapsing set partitions (Hardy 2006). If the members 1, 2, 3 of the set {1, 2, 3, 4, 5} are made indistinguishable from each other and called *a*, and 4 and 5 are made indistinguishable from each other and called *b*, then the set {1, 2, 3, 4, 5} has “collapsed” to the multiset $M = \{a, a, a, b, b\}$. Therefore the subdivisions of M can be recovered using the same substitution in the partitions of {1, 2, 3, 4, 5}. For example, S_1 in (7) can be recovered from $\{\{1, 4\}, \{2, 5\}, \{3\}\}$ or $\{\{3, 5\}, \{2, 4\}, \{1\}\}$ and so on. As this last example shows, a subdivision might correspond to several partitions. The number of partitions corresponding to the same subdivision can be computed using the countP function of the package. However, to find multi-index partitions using set partitions is not a particularly efficient algorithm since the computational cost is proportional to the n -th Bell number, if n is the sum of the multi-index components (Charalambides 2002).

The mkMSet function is based on a different strategy which takes into account the partitions of the multi-index components. When $m = 1$, the mkMSet function lists all the partitions λ of the integer i . Recall that a partition of an integer i is a sequence $\lambda = (\lambda_1, \lambda_2, \dots)$ of weakly decreasing positive integers, named parts of λ , such that $\lambda_1 + \lambda_2 + \dots = i$. A different notation is $\lambda = (1^{r_1}, 2^{r_2}, \dots)$, where r_1, r_2, \dots are the number of parts of λ equal to 1, 2, ... respectively. The length of the partition is $l(\lambda) = r_1 + r_2 + \dots$. We write $\lambda \vdash i$ to denote that λ is a partition of i . In the following, we describe the main steps of the mkMSet function by working on an example.

Suppose we want to generate all the partitions of (3, 2). First consider the partitions of (3, 0) obtained from the partitions (3), (1, 2), (1³) of the integer 3, and the partitions of (0, 2) obtained from the partitions (2), (1²) of the integer 2, that is

$$\Lambda_1 = \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \Lambda_2 = \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix}, \Lambda_3 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \vdash \begin{pmatrix} 3 \\ 0 \end{pmatrix} \tag{9}$$

$$\Lambda_4 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \Lambda_5 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \vdash \begin{pmatrix} 0 \\ 2 \end{pmatrix}. \tag{10}$$

The following iterated adding-appending rule is thus implemented.

1. Consider the partition Λ_5 in (10).

1.1 Add the first column of Λ_5 to each column of Λ_1, Λ_2 and Λ_3 in (9) one by one with the following rules: the sum must be done only once (if the column has multiplicities greater than one) taking as reference the first column; the sum can be done only to columns whose second component is zero and without subsequent elements (in the same row) greater than or equal to the integer we are adding. Then we have

$$\Lambda_1^{(1,1)} = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \quad \Lambda_2^{(1,1)} = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix} \quad \Lambda_3^{(1,1)} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}. \tag{11}$$

1.2 Append the same column to each partition Λ_1, Λ_2 and Λ_3 in (10), that is

$$\Lambda_1^{(1,2)} = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \quad \Lambda_2^{(1,2)} = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \Lambda_3^{(1,2)} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{12}$$

1.3 Repeat steps 1.1 and 1.2 for the second column of Λ_5 with respect to the partitions generated in (11) and (12):

$\Lambda_1^{(1,1)} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$	add \Rightarrow rule out	append $\Rightarrow \begin{pmatrix} 3 & 0 \\ 1 & 1 \end{pmatrix}$
$\Lambda_2^{(1,1)} = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}$	add $\Rightarrow \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}$	append $\Rightarrow \begin{pmatrix} 1 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix}$
$\Lambda_2^{(2,1)} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$	add \Rightarrow rule out	append $\Rightarrow \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{pmatrix}$
$\Lambda_3^{(1,1)} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	add $\Rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$	append $\Rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$
$\Lambda_1^{(1,2)} = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}$	add \Rightarrow rule out	append $\Rightarrow \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$
$\Lambda_2^{(1,2)} = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	add \Rightarrow rule out	append $\Rightarrow \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$
$\Lambda_3^{(1,2)} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	add \Rightarrow rule out	append $\Rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$

2. Repeat step 1 for Λ_4 in (10):

$$\begin{aligned} \Lambda_1 &= \begin{pmatrix} 3 \\ 0 \end{pmatrix} & \text{add} &\Rightarrow \begin{pmatrix} 3 \\ 2 \end{pmatrix} & \text{append} &\Rightarrow \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \\ \Lambda_2 &= \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix} & \text{add} &\Rightarrow \begin{pmatrix} 1 & 2 \\ 2 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 0 & 2 \end{pmatrix} & \text{append} &\Rightarrow \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \\ \Lambda_3 &= \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} & \text{add} &\Rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 2 & 0 & 0 \end{pmatrix} & \text{append} &\Rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \end{aligned}$$

More generally, the `mkmSet` function lists all the partitions $\Lambda \vdash i$, with the columns reordered in increasing lexicographic order, together with the number of set partitions corresponding to the same multi-index partition, that is $i!/\Lambda!m(\Lambda)!$. In the latest version of the `kStatistics` package, among the input parameters of the `mkmSet` function, an input flag parameter has been inserted aiming to print the multi-index partitions in a more compact form. See the following example.

Example 2: To get all the partitions of (2, 1) run

```
> mkmSet(c(2,1), TRUE)
[( 0 1 )( 1 0 )( 1 0 ), 1 ]
[( 0 1 )( 2 0 ), 1 ]
[( 1 0 )( 1 1 ), 2 ]
[( 2 1 ), 1 ]
```

Note that the integers 1, 1, 2, 1 correspond to the coefficients $2!1!/\Lambda!m(\Lambda)!$.

Example 3: To get all the partitions of the integer 3 run

```
> mkmSet(c(3), TRUE)
[( 1 )( 1 )( 1 ), 1 ]
[( 1 )( 2 ), 3 ]
[( 3 ), 1 ]
```

The `mkmSet` function is called by the `intPart` function, specifically designed with the purpose of listing only all the partitions of a given integer in increasing order. The input flag parameter allows us to print the partitions in a more compact form.

Example 4: To get all the partitions of the integer 4 run

```
> intPart(4, TRUE)
[ 1 1 1 1 ]
[ 1 1 2 ]
[ 2 2 ]
[ 1 3 ]
[ 4 ]
```

The `parts` function of the `partitions` package (Hankin 2006) lists all the partitions of a given integer, but in decreasing order. Instead the `get.partitions` function of the `nilde` package (Arnqvist et al. 2021) finds all the partitions of a given integer with a fixed length $l(\lambda)$ (Voinov and Pya Arnqvist 2017). If $l(\lambda)$ is equal to the given integer, the `get.partitions` function lists all the partitions in increasing order.

3 kStatistics

The i -th k -statistic κ_i is the (unique) symmetric estimator whose expectation is the i -th cumulant $k_i(Y)$ of a population character Y and whose variance is a minimum relative to all other unbiased estimators.

The `nKS` function generates the numerical value of the i -th k -statistic starting from a data sample. The computation relies on the following polynomials

$$P_t(y) = \sum_{j=1}^t y^j S(t, j) (-1)^{j-1} (j-1)! \quad \text{for } t = 1, \dots, i \tag{13}$$

where $\{S(t, j)\}$ are the Stirling numbers of second kind, generated through the `nStirling2` function. In detail, suppose to have a sample $\{a_1, \dots, a_N\}$ of N numerical data and denote with p_t the t -th power sum in the numerical data

$$p_t(a_1, \dots, a_N) = \sum_{j=1}^N a_j^t, \quad \text{for } t \geq 1. \tag{14}$$

To carry out the numerical value of the i -th k -statistic for $i \leq N$, the nKS function computes the explicit expression of the polynomial of degree i

$$Q_i(y) = \sum_{\lambda \vdash i} d_\lambda \mathcal{P}_\lambda(y) p_\lambda \tag{15}$$

where the sum is over all the partitions $\lambda = (1^{r_1}, 2^{r_2}, \dots) \vdash i$, and

$$d_\lambda = \frac{i!}{(1!)^{r_1} r_1! (2!)^{r_2} r_2! \dots} \quad \mathcal{P}_\lambda(y) = [\mathcal{P}_1(y)]^{r_1} [\mathcal{P}_2(y)]^{r_2} \dots \quad p_\lambda = [p_1]^{r_1} [p_2]^{r_2} \dots \tag{16}$$

with $\{\mathcal{P}_t(y)\}$ and $\{p_t\}$ given in (13) and (14) respectively. The final step is to replace the powers y^t in the explicit form of the polynomial (15) with $(-1)^{t-1} (t-1)! / (N)_t$ for $t = 1, \dots, i$.

The main steps of the nKS function are summarized in the following.

Function nKS

- i) Compute the power sums p_t in (14) for $t = 1, \dots, i$.
- ii) Compute $S(t, j) (-1)^{j-1} (j-1)!$ in (13) for $j = 1, \dots, t$ and $t = 1, \dots, i$.
- iii) Using the `mkmSet` function, compute all the partitions $\lambda \vdash i$.
- iv) For a given partition λ , expand the product $\mathcal{P}_\lambda(y)$ in (15) and compute the coefficient $d_\lambda p_\lambda$ of each monomial in $Q_i(y)$ using (16).
- v) For $t = 1, \dots, i$ multiply $(-1)^{t-1} (t-1)! / (N)_t$ with the coefficients of the monomial of degree t carried out at the previous step and do the sum over all the resulting numerical values.
- vi) Repeat steps iv) and v) for all the partitions λ carried out at step iii) and do the sum over all the resulting numerical values.

Example 5: Using (15) for $i = 1$, we have $Q_1(y) = \mathcal{P}_1(y) p_1 = y \sum_{j=1}^N a_j$ and plugging $1/N$ in place of y , the sample mean is recovered. Using (15) for $i = 2$, we have

$$Q_2(y) = \mathcal{P}_2(y) p_2 + (\mathcal{P}_1(y) p_1)^2 = y \sum_{j=1}^N a_j^2 + y^2 \left(\left(\sum_{j=1}^N a_j \right)^2 - \sum_{j=1}^N a_j^2 \right)$$

and plugging $1/N$ in place of y and $-1/N(N-1)$ in place of y^2 , the sample variance is recovered. Compare the values of the sample mean, computed with the nKS function and the `mean` function, and the sample variance, computed with the nKS function and the `var` function, for the following dataset:

```
> data<-c(16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68,
16.08, 19.43, 8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10, 15.02,
16.83, 16.98, 19.92, 9.47, 11.68, 13.41, 15.35, 19.11)
> nKS(1,data)
[1] 14.02167
> mean(data)
[1] 14.02167
> nKS(2,data)
[1] 12.65007
> var(data)
[1] 12.65007
```

Using the nKS function, for instance, the sample skewness and the sample kurtosis can be computed. Let us recall that the sample skewness is a measure of the central tendency of a univariate sample and can be computed as $\kappa_3 / \kappa_2^{3/2}$ where κ_2 and κ_3 are the second and the third k -statistics respectively (Joanes and Gill 1998). The sample kurtosis is a measure of the tail-heaviness of a sample distribution. The sample excess kurtosis is defined as the sample kurtosis minus 3 and can be computed as κ_4 / κ_2^2 where κ_2 and κ_4 are the second and the fourth k -statistics respectively (Joanes and Gill 1998).

```
> nKS(3,data)/sqrt(nKS(2,data))^(3/2)
[1] -0.03216229
> nKS(4,data)/nKS(2,data)^2 + 3
[1] 2.114708
```

A similar strategy is employed to compute multivariate k -statistics (the nKM function) of a sample data matrix whose columns each represent a population character. To simplify the notation, in the following we deal with the case of a bivariate data set $\{(a_{1,1}, a_{2,1}) \dots, (a_{1,N}, a_{2,N})\}$ of N paired numerical data. Denote with $p_{(s,t)}$ the bivariate power sum in the paired data

$$p_{(s,t)}[(a_{1,1}, a_{2,1}), \dots, (a_{1,N}, a_{2,N})] = \sum_{j=1}^N a_{1,j}^s a_{2,j}^t \quad \text{for } s, t \geq 1. \tag{17}$$

Suppose $i = (i_1, i_2)$ with $i_1, i_2 \leq N$ and set $i = i_1 + i_2$. To carry out the numerical value of the i -th multivariate k -statistic, the nKM function finds the explicit expression of the polynomial

$$Q_i(y) = \sum_{\Lambda \vdash i} d_{\Lambda} P_{\Lambda}(y) p_{\Lambda} \tag{18}$$

where the sum is over all the partitions $\Lambda = (\lambda_1^{r_1}, \lambda_2^{r_2}, \dots) \vdash i$, and

$$d_{\Lambda} = \frac{i!}{\Lambda! \mathbf{m}(\Lambda)!} P_{\Lambda}(y) = [\mathcal{P}_{|\lambda_1|}(y)]^{r_1} [\mathcal{P}_{|\lambda_2|}(y)]^{r_2} \dots \quad p_{\Lambda} = [p_{\lambda_1}]^{r_1} [p_{\lambda_2}]^{r_2} \dots \tag{19}$$

with $\{\mathcal{P}_t(y)\}$ and $\{p_{(s,t)}\}$ given in (13) and (17) respectively. As for the univariate k -statistics, the final step consists in replacing the powers y^j in the explicit expression of the polynomial (18) with the numerical values $(-1)^{j-1} (j-1)! / (N)_j$ for $j = 1, \dots, i$.

The main steps of the nKM function are summarized in the following.

Function nKM

- i) Compute the bivariate power sums $p_{(s,t)}$ in (17) for $s = 1, \dots, i_1$ and $t = 1, \dots, i_2$.
 - ii) For $i = i_1 + i_2$, compute $S(t, j) (-1)^{j-1} (j-1)!$ in (13) for $j = 1, \dots, t$ and $t = 1, \dots, i$.
 - iii) Using the `mkMSet` function, compute all the partitions $\Lambda \vdash i$.
 - iv) For a given partition Λ , expand the product $P_{\Lambda}(y)$ in (18) and compute the coefficient $d_{\Lambda} p_{\Lambda}$ of each monomial in $Q_i(y)$ using (19).
 - v) For $j = 1, \dots, i$, multiply $(-1)^{j-1} (j-1)! / (N)_j$ with the coefficient of the monomial of degree j carried out at the previous step and do the sum over all the resulting numerical values.
 - vi) Repeat steps iv) and v) for all the partitions Λ carried out at step iii) and do the sum over all the resulting numerical values.
-

Example 6: To estimate the joint cumulant $c_{2,1}$ of the following dataset, run

```
> data1<-list(c(5.31, 11.16), c(3.26, 3.26), c(2.35, 2.35), c(8.32, 14.34),
c(13.48, 49.45), c(6.25, 15.05), c(7.01, 7.01), c(8.52, 8.52), c(0.45, 0.45),
c(12.08, 12.08), c(19.39, 10.42))
> nKM(c(2, 1), data1)
[1] -23.7379
```

If the first column are observations of a population character X and the second column observations of a population character Y , then $c_{2,1}$ measures how far from connectedness (as opposite to independence) are X^2 and Y (E. Di Nardo, Marena, and Semeraro 2020). A similar meaning has the estimation of the joint cumulant $c_{2,2,2}$ of the following dataset:

```
> data2<-list(c(5.31, 11.16, 4.23), c(3.26, 3.26, 4.10), c(2.35, 2.35, 2.27),
c(4.31, 10.16, 6.45), c(3.1, 2.3, 3.2), c(3.20, 2.31, 7.3))
> nKM(c(2, 2, 2), data2)
[1] 678.1045
```

4 Polykays

Similarly to k -statistics, polykays are symmetric unbiased estimators of cumulant products. More in detail, when evaluated on a random sample, the i -th polykay gives an estimation of the product $k_{i_1}(Y) \cdots k_{i_m}(Y)$, where $\mathbf{i} = (i_1, \dots, i_m) \in \mathbb{N}_0^m$ and $\{k_{i_j}(Y)\}$ are cumulants of a population character Y .

To simplify the notation, in the following we show how to compute the i -th polykay of N numerical data $\{a_1, \dots, a_N\}$ using the nPS function for $\mathbf{i} = (i_1, i_2)$. Set $i = i_1 + i_2$ and suppose $i \leq N$. The computation relies on the so-called logarithmic polynomials

$$\tilde{P}_t(y_1, \dots, y_i) = \sum_{\lambda \vdash t} y_\lambda d_\lambda (-1)^{l(\lambda)-1} (l(\lambda) - 1)! \tag{20}$$

for $t = 1, \dots, i$ where the sum is over all the partitions $\lambda = (1^{r_1}, 2^{r_2}, \dots) \vdash t$, d_λ is given in (16) and $y_\lambda = y_1^{r_1} y_2^{r_2} \cdots$. To compute the polykay of order (i_1, i_2) , the nPS function finds the explicit expression of the polynomial

$$A_i(y_1, \dots, y_i) = \sum_{\lambda \vdash i} d_\lambda \tilde{P}_\lambda(y_1, \dots, y_i) p_\lambda \tag{21}$$

where the sum is over all the partitions $\lambda = (1^{r_1}, 2^{r_2}, \dots) \vdash i$, d_λ and p_λ are given in (16) and

$$\tilde{P}_\lambda(y_1, \dots, y_i) = [\tilde{P}_1(y_1, \dots, y_i)]^{r_1} [\tilde{P}_2(y_1, \dots, y_i)]^{r_2} \cdots$$

with $\{\tilde{P}_t(y_1, \dots, y_i)\}$ given in (20). Note that the monomials in $A_i(y_1, \dots, y_i)$ are of type $y_\lambda = y_1^{r_1} y_2^{r_2} \cdots$ with $\lambda = (1^{r_1}, 2^{r_2}, \dots) \vdash i$. The final step is to plug suitable numerical values in place of y_λ depending on how the partition λ is constructed. Indeed, set

$$\tilde{q}(i_1, i_2) = \left\{ \lambda' + \lambda'' \vdash i \mid \lambda' = (1^{s_1}, 2^{s_2}, \dots) \vdash i_1, \lambda'' = (1^{t_1}, 2^{t_2}, \dots) \vdash i_2 \right\} \tag{22}$$

where $\lambda' + \lambda'' = (1^{r_1}, 2^{r_2}, \dots)$ with $r_j = s_j + t_j$ for $j = 1, 2, \dots$. Then y_λ is replaced by 0 if $\lambda \notin \tilde{q}(i_1, i_2)$ otherwise by

$$\frac{(-1)^{l(\lambda')-1} (l(\lambda') - 1)! (-1)^{l(\lambda'')-1} (l(\lambda'') - 1)! d_{\lambda'} d_{\lambda''}}{(N)_{l(\lambda') + l(\lambda'')}} \frac{d_{\lambda'} d_{\lambda''}}{d_{\lambda' + \lambda''}} \tag{23}$$

Note that $d_{\lambda'}$ and $d_{\lambda''}$ in (23) are recovered from (16).

The main steps of the nPS function are summarized in the following.

Function nPS

- i) Set $i = i_1 + i_2$ and compute the power sums p_t in (14) for $t = 1, \dots, i$.
 - ii) Generate the polynomials $\tilde{P}_t(y_1, \dots, y_i)$ in (20) for $t = 1, \dots, i$.
 - iii) Using the `mkmSet` function, compute all the partitions $\lambda \vdash i$.
 - iv) For a given partition λ , expand the product $\tilde{P}_\lambda(y_1, \dots, y_i)$ in (21); then plug (23) or 0 in each monomial y_λ , depending if λ is or not in the set $\tilde{q}(i_1, i_2)$ given in (22).
 - v) Multiply the numerical value of \tilde{P}_λ carried out at step iv) with $d_\lambda p_\lambda$ given in (16).
 - vi) Repeat steps iv) and v) for all the partitions λ carried out at step iii) and do the sum over all the resulting numerical values.
-

Example 7: Suppose we need to estimate the square of the variance σ^2 of the population character Y from which the data of Example 5 have been sampled. We have

```
> nKS(2, data)^2
[1] 160.0243
> var(data)^2
[1] 160.0243
```

but k_2^2 is not an unbiased estimator of the square of σ^2 . An unbiased estimator of such a square is the polykay of order $(2, 2)$, that is

```
> nPS(c(2, 2), data)
[1] 154.1177
```

Multivariate polykeys are unbiased estimators of products of multivariate cumulants and the nPM function returns a numerical value for these estimators when evaluated on a random sample. As before, to show how the nPM function works, we consider a bivariate sample of N numerical data, that is $\{(a_{1,1}, a_{2,1}) \dots (a_{1,N}, a_{2,N})\}$. If we choose $\mathbf{i} = (i_1, i_2)$ and $\mathbf{j} = (j_1, j_2)$ with $i_1 + i_2 + j_1 + j_2 \leq N$ as input of the nPM function, the output is a numerical value which represents an estimated value of the product $k_i(X, Y)k_j(X, Y)$, where $k_i(X, Y)$ and $k_j(X, Y)$ are cumulants of the population characters (X, Y) . The computation relies on suitable polynomials in the indeterminates $\{y_{(s,t)}\}$ for $s = 0, \dots, w_1, t = 0, \dots, w_2$, with $s + t > 0$ and $w_1 = i_1 + j_1, w_2 = i_2 + j_2$. These polynomials are a multivariable generalization of (20), that is

$$\tilde{P}_k(\{y_{(s,t)}\}) = \sum_{\Lambda \vdash k} y_\Lambda d_\Lambda (-1)^{l(\Lambda)-1} (l(\Lambda) - 1)! \tag{24}$$

for $0 < k \leq w = (w_1, w_2)$, where the sum is over all the partitions $\Lambda = (\lambda_1^{r_1}, \lambda_2^{r_2}, \dots) \vdash k$ and $y_\Lambda = y_{\lambda_1}^{r_1} y_{\lambda_2}^{r_2} \dots$. To compute the multivariate polykey of order (\mathbf{i}, \mathbf{j}) , the nPM function finds the explicit expression of the polynomial

$$\mathcal{A}_w(\{y_{(s,t)}\}) = \sum_{\Lambda \vdash w} d_\Lambda \tilde{P}_\Lambda(\{y_{(s,t)}\}) p_\Lambda \tag{25}$$

where the sum is over all the partitions $\Lambda = (\lambda_1^{r_1}, \lambda_2^{r_2}, \dots) \vdash w$, d_Λ and p_Λ are given in (19), and

$$\tilde{P}_\Lambda(\{y_{(s,t)}\}) = [\tilde{P}_{\lambda_1}(\{y_{(s,t)}\})]^{r_1} [\tilde{P}_{\lambda_2}(\{y_{(s,t)}\})]^{r_2} \dots$$

with $\{\tilde{P}_\lambda(\{y_{(s,t)}\})\}$ given in (24). The monomials in $\mathcal{A}_w(\{y_{(s,t)}\})$ are of type y_Λ with $\Lambda \vdash w$. The final step is to plug suitable numerical values in place of y_Λ depending on how the partition Λ is constructed. Indeed, set

$$\tilde{q}(w) = \left\{ \Lambda' + \Lambda'' \vdash w \mid \Lambda' = (\lambda_1^{s_1}, \lambda_2^{s_2}, \dots) \vdash \mathbf{i}, \Lambda'' = (\lambda_1^{t_1}, \lambda_2^{t_2}, \dots) \vdash \mathbf{j} \right\}, \tag{26}$$

where $\Lambda' + \Lambda'' = (\tilde{\lambda}_1^{r_1}, \tilde{\lambda}_2^{r_2}, \dots)$ is built with the columns of Λ' and Λ'' rearranged in increasing lexicographic order and such that $r_j = s_j$ if $\tilde{\lambda}_j = \lambda_j'$ or $r_j = t_j$ if $\tilde{\lambda}_j = \lambda_j''$ or $r_j = s_j + t_j$ if $\tilde{\lambda}_j = \lambda_j' = \lambda_j''$. Therefore in the explicit expression of (25), y_Λ is replaced by 0 if $\Lambda \notin \tilde{q}(w)$ otherwise by

$$\frac{(-1)^{l(\Lambda')-1} (l(\Lambda') - 1)! (-1)^{l(\Lambda'')-1} (l(\Lambda'') - 1)! d_{\Lambda'} d_{\Lambda''}}{(N)_{l(\Lambda') + l(\Lambda'')}} \frac{d_{\Lambda'} d_{\Lambda''}}{d_{\Lambda' + \Lambda''}}. \tag{27}$$

Note that $d_{\Lambda'}$ and $d_{\Lambda''}$ in (27) are recovered from (19).

The main steps of the nPM function are summarized in the following.

Function nPM

- i) Set $w_1 = i_1 + j_1$ and $w_2 = i_2 + j_2$; compute the power sums $p_{(s,t)}$ in (17) for $s = 1, \dots, w_1$ and $t = 1, \dots, w_2$.
 - ii) Generate the polynomials $\tilde{P}_k(\{y_{(s,t)}\})$ in (24) for $0 < k \leq w = (w_1, w_2)$.
 - iii) Using the `mkmSet` function, compute all the partitions $\Lambda \vdash w$.
 - iv) For a given partition Λ , expand the product $\tilde{P}_\Lambda(\{y_{(s,t)}\})$ in (25) and plug (27) or 0 in each obtained monomial of type y_Λ depending if Λ is or not in $\tilde{q}(w)$ given in (26).
 - v) Multiply the numerical value of \tilde{P}_Λ obtained at step iv) with $d_\Lambda p_\Lambda$ given in (19).
 - vi) Repeat steps iv) and v) for all the partitions Λ carried out at step iii) and do the sum over all the resulting numerical values.
-

Example 8: For the same dataset employed in Example 6, to estimate $k_{(2,1)}(X, Y)k_{(1,0)}(X, Y)$ run

```
> nPM(list(c(2,1), c(1,0)), data1)
[1] 48.43243
```

Remark 1: The master `nPolyk` function runs one of the `nKS`, `nKM`, `nPS` and `nPM` functions depending if we ask for simple k -statistics, multivariate k -statistics, simple polykeys or multivariate polykeys.

5 Bell polynomials and generalizations

The algorithms to produce k -statistics and polykays rely on handling suitable polynomial families which are special cases of generalizations of Bell polynomials, as introduced in this section. Moreover, there are further families of polynomials widely used in applications which are special cases of these polynomials. For the most popular ones, we have implemented special functions in the **kStatistics** package. The list is not exhaustive, see for instance Roman (1984). Furthermore additional families of polynomials might be recovered using the multivariate Faà di Bruno's formula. We will give some examples in the next section.

The i -th generalized (complete exponential) Bell polynomial in the indeterminates y_1, \dots, y_n is

$$h_i(y_1, \dots, y_n) = i! \sum_{\substack{\Lambda \vdash s_1, \dots, \tilde{\Lambda} \vdash s_n \\ s_1 + \dots + s_n = i}} y_1^{l(\Lambda)} \dots y_n^{l(\tilde{\Lambda})} \frac{g_{1,\Lambda} \dots g_{n,\tilde{\Lambda}}}{\Lambda! \dots \tilde{\Lambda}! m(\Lambda)! \dots m(\tilde{\Lambda})!} \tag{28}$$

where the sum is over all the partitions $\Lambda \vdash s_1, \dots, \tilde{\Lambda} \vdash s_n$ with s_1, \dots, s_n m -tuples of non-negative integers such that $s_1 + \dots + s_n = i$ and

$$\begin{aligned} g_{1,\Lambda} &= g_{1,\lambda_1}^{r_1} g_{1,\lambda_2}^{r_2} \dots && \text{for } \Lambda = (\lambda_1^{r_1}, \lambda_2^{r_2}, \dots) \\ &\vdots \\ g_{n,\tilde{\Lambda}} &= g_{n,\tilde{\lambda}_1}^{t_1} g_{n,\tilde{\lambda}_2}^{t_2} \dots && \text{for } \tilde{\Lambda} = (\tilde{\lambda}_1^{t_1}, \tilde{\lambda}_2^{t_2}, \dots) \end{aligned} \tag{29}$$

with $\{g_{1,\lambda}\}, \dots, \{g_{n,\tilde{\lambda}}\}$ multi-indexed sequences. These polynomials are the output of the `GCBellPol` function.

Example 9: To get $h_{(1,1)}(y_1, y_2)$ run

```
> GCBellPol(c(1,1), 2)
[1] (y1)(y2)g1[0,1]g2[1,0] + (y1)(y2)g1[1,0]g2[0,1] + (y1^2)g1[0,1]g1[1,0] +
(y1)g1[1,1] + (y2^2)g2[0,1]g2[1,0] + (y2)g2[1,1]
```

The `e_GCBellPol` function evaluates $h_i(y_1, \dots, y_n)$ when its indeterminates y_1, \dots, y_n and/or its coefficients are substituted with numerical values.

Example 10: To plug the values from 1 to 6 respectively into the coefficients `g1[,]` and `g2[,]` of the polynomial $h_{(1,1)}(y_1, y_2)$ given in Example 9 run

```
> e_GCBellPol(c(1,1), 2, "g1[0,1]=1, g1[1,0]=2, g1[1,1]=3, g2[0,1]=4, g2[1,0]=5,
g2[1,1]=6")
[1] 13(y1)(y2) + 2(y1^2) + 3(y1) + 20(y2^2) + 6(y2)
```

To evaluate $h_{(1,1)}(1, 5)$ run

```
> e_GCBellPol(c(1,1), 2, "y1=1, y2=5, g1[0,1]=1, g1[1,0]=2, g1[1,1]=3, g2[0,1]=4,
g2[1,0]=5, g2[1,1]=6")
[1] 600
```

When the multi-indexed sequences $\{g_{1,\lambda}\}, \dots, \{g_{n,\tilde{\lambda}}\}$ are all equal, the number of distinct addends in (28) might reduce and the corresponding generalized Bell polynomial is denoted by $\tilde{h}_i(y_1, \dots, y_n)$. To deal with this special case, we have inserted an input flag parameter in the `e_GCBellPol` function.

Example 11: To compare $\tilde{h}_{(1,1)}(y_1, y_2)$ with $h_{(1,1)}(y_1, y_2)$ given in Example 9 run

```
> GCBellPol(c(1,1), 2, TRUE)
[1] 2(y1)(y2)g[0,1]g[1,0] + (y1^2)g[0,1]g[1,0] + (y1)g[1,1] + (y2^2)g[0,1]g[1,0] +
(y2)g[1,1]
```

Set $n = 1$ in (28). Then $h_i(y_1, \dots, y_n)$ reduces to the univariate polynomial

$$h_i(y) = \sum_{\Lambda \vdash i} y^{l(\Lambda)} d_\Lambda g_\Lambda \tag{30}$$

where the sum is over all the partitions $\Lambda = (\lambda_1^{r_1}, \lambda_2^{r_2}, \dots) \vdash i$, d_Λ is given in (19) and $g_\Lambda = g_{\lambda_1}^{r_1} g_{\lambda_2}^{r_2} \dots$.

Example 12: To get $h_{(1,1)}(y)$ run

```
> GCBellPol(c(1,1),1)
[1] (y^2)g[0,1]g[1,0] + (y)g[1,1]
```

Remark 2: For all $i \in \mathbb{N}_0^m$, we have $h_i(y_1 + \dots + y_n) = \tilde{h}_i(y_1, \dots, y_n)$, where $\tilde{h}_i(y_1, \dots, y_n)$ is the i -th generalized Bell polynomial (28) corresponding to all equal multi-indexed sequences $\{g_{1,\lambda}\}, \dots, \{g_{n,\lambda}\}$ (Elvira Di Nardo 2011). Therefore the `e_GCBellPol` function, with the input flag `TRUE`, produces also an explicit expression of $h_i(y_1 + \dots + y_n)$.

The algorithm to generate joint moments in terms of joint cumulants and vice versa follows the same pattern designed to generate $\{h_i(y)\}$. Indeed if $\{k_i(\mathbf{Y})\}$ and $\{m_i(\mathbf{Y})\}$ denote the sequences of joint cumulants and joint moments of a random vector \mathbf{Y} respectively, then

$$m_i(\mathbf{Y}) = \sum_{\Lambda \vdash i} d_\Lambda k_\Lambda(\mathbf{Y}) \text{ and } k_i(\mathbf{Y}) = \sum_{\Lambda \vdash i} (-1)^{l(\Lambda)-1} (l(\Lambda) - 1)! d_\Lambda m_\Lambda(\mathbf{Y}), \tag{31}$$

where the sum is over all the partitions $\Lambda = (\lambda_1^{r_1}, \lambda_2^{r_2}, \dots) \vdash i$, d_Λ is given in (19) and

$$m_\Lambda(\mathbf{Y}) = [m_{\lambda_1}(\mathbf{Y})]^{r_1} [m_{\lambda_2}(\mathbf{Y})]^{r_2} \dots \quad k_\Lambda(\mathbf{Y}) = [k_{\lambda_1}(\mathbf{Y})]^{r_1} [k_{\lambda_2}(\mathbf{Y})]^{r_2} \dots$$

In particular

- the `mom2cum` function returns the right hand side of the first equation in (31), using the same algorithm producing $h_i(y)$ in (30) with the sequence $\{k_\lambda\}$ in place of $\{g_\lambda\}$ and with 1 in place of y ;
- the `cum2mom` function returns the right hand side of the latter equation in (31), using the same algorithm producing $h_i(y)$ in (30) with the sequence $\{m_\lambda\}$ in place of $\{g_\lambda\}$ and with $(-1)^{j-1} (j-1)!$ in place of the powers y^j for $j = 1, \dots, |i|$.

When the multi-index i reduces to an integer i , formulae (31) are the classical expressions of univariate moments in terms of univariate cumulants and vice versa. The `mom2cum` and `cum2mom` functions do the same when the input is an integer.

Example 13: To get $m_{(3,1)}$ in terms of $k_{(i,j)}$ run

```
> mom2cum(c(3,1))
[1] k[0,1]k[1,0]^3 + 3k[0,1]k[1,0]k[2,0] + k[0,1]k[3,0] + 3k[1,0]^2k[1,1] +
3k[1,0]k[2,1] + 3k[1,1]k[2,0] + k[3,1]
```

To get $k_{(3,1)}$ in terms of $m_{(i,j)}$ run

```
> cum2mom(c(3,1))
[1] - 6m[0,1]m[1,0]^3 + 6m[0,1]m[1,0]m[2,0] - m[0,1]m[3,0] +
6m[1,0]^2m[1,1] - 3m[1,0]m[2,1] - 3m[1,1]m[2,0] + m[3,1]
```

Remark 3: There are different functions in R performing similar computations for cumulants and moments: for instance see De Leeuw, J. (2012) for the multivariate case. A different strategy would rely on the recursive relations between cumulants and moments (Domino, Gawron, and Pawela 2018).

Similarly to (31), some of the polynomials employed in the previous sections are generated using the same pattern developed to find the explicit expression of $h_i(y)$ in (30):

- The generation of an explicit expression of $Q_i(y)$ in (18) parallels the one implemented for $h_i(y)$ with 1 in place of y and with the polynomial sequence $\{\mathcal{P}_{|\lambda|}(y)p_\lambda\}$ in place of the sequence $\{g_\lambda\}$;
- the same for the polynomials $\tilde{P}_k(\{y_{(s,t)}\})$ in (24) with $(-1)^{j-1} (j-1)!$ for $j = 1, \dots, |i|$ in place of the powers y^j and with the polynomial sequence $\{y_\lambda\}$ in place of the sequence $\{g_\lambda\}$;
- the same for the polynomials $\mathcal{A}_w(\{y_{(s,t)}\})$ in (25) with 1 in place of y and with the polynomial sequence $\{\tilde{P}_\lambda(\{y_{(s,t)}\})p_\lambda\}$ in place of the sequence $\{g_\lambda\}$.

Note that when the multi-index i in (30) reduces to a positive integer i , then the polynomial $h_i(y)$ becomes

$$h_i(y) = \sum_{\lambda \vdash i} d_\lambda y^{l(\lambda)} g_\lambda \tag{32}$$

where the sum is over all the partitions $\lambda = (1^{r_1}, 2^{r_2}, \dots) \vdash i$, d_λ is given in (16) and $g_\lambda = g_1^{r_1} g_2^{r_2} \dots$ with $\{g_j\}$ a suitable sequence.

Example 14: To get $h_3(y)$ run

```
> GCBellPol(c(3), 1)
[1] (y^3)g[1]^3 + 3(y^2)g[1]g[2] + (y)g[3]
```

With a combinatorial structure very similar to (32), the i -th general partition polynomial has the following expression in the indeterminates y_1, \dots, y_i

$$G_i(a_1, \dots, a_i; y_1, \dots, y_i) = \sum_{\lambda \vdash i} d_\lambda a_{l(\lambda)} y_\lambda \tag{33}$$

where the sum is over all the partitions $\lambda = (1^{r_1}, 2^{r_2}, \dots) \vdash i$, d_λ is given in (16), $\{a_j\}$ is a suitable numerical sequence and $y_\lambda = y_1^{r_1} y_2^{r_2} \dots$. It's a straightforward exercise to prove that

$$G_i(a_1, \dots, a_i; y_1, \dots, y_i) = \sum_{j=1}^i a_j B_{i,j}(y_1, \dots, y_{i-j+1}), \tag{34}$$

where $\{B_{i,j}\}$ are the (partial) exponential Bell polynomials

$$B_{i,j}(y_1, \dots, y_{i-j+1}) = \sum_{\bar{p}(i,j)} d_\lambda y_\lambda \tag{35}$$

where $\bar{p}(i,j) = \{\lambda = (1^{r_1}, 2^{r_2}, \dots) \vdash i | l(\lambda) = j\}$, d_λ is given in (16) and $y_\lambda = y_1^{r_1} y_2^{r_2} \dots$. The polynomials in (33) are widely used in applications such as combinatorics, probability theory and statistics (Charalambides 2002). As particular cases, they include the exponential polynomials and their inverses, the logarithmic polynomials (20), the potential polynomials and many others (Roman 1984). The general partition polynomials are the output of the gpPart function.

Example 15: To get $G_4(a_1, a_2, a_3, a_4; y_1, y_2, y_3, y_4)$ run

```
> gpPart(4)
[1] a4(y1^4) + 6a3(y1^2)(y2) + 3a2(y2^2) + 4a2(y1)(y3) + a1(y4)
```

When $a_1 = \dots = a_i = 1$, the i -th general partition polynomial in (34) reduces to the complete (exponential) Bell polynomial

$$G_i(1, \dots, 1; y_1, \dots, y_i) = \sum_{j=1}^i B_{i,j}(y_1, \dots, y_{i-j+1}) \tag{36}$$

where $\{B_{i,j}\}$ are the (partial) exponential Bell polynomials (35). For instance, the polynomial $Q_i(y)$ in (15) is generated using the same pattern developed to generate (36) with $\mathcal{P}_j(y)p_j$ in place of y_j .

The eBellPol function returns the complete (exponential) Bell polynomials (36). The same function also produces the (partial) exponential Bell polynomial $B_{i,j}(y_1, \dots, y_{i-j+1})$ using (33) with $a_k = \delta_{k,j}$ (the Kronecker delta) for $k = 1, \dots, i$. Mihoubi (2008) gives a rather extensive survey of applications of these homogeneous polynomials.

Example 16: To get $B_{5,3}(y_1, y_2, y_3)$ run

```
> eBellPol(5, 3)
[1] 15(y1)(y2^2) + 10(y1^2)(y3)
```

To get $G_4(1, 1, 1, 1; y_1, y_2, y_3, y_4)$ run

```
> eBellPol(4)
[1] (y1^4) + 6(y1^2)(y2) + 3(y2^2) + 4(y1)(y3) + (y4)
```

The oBellPol function returns the partial (ordinary) Bell polynomials

$$\hat{B}_{i,j}(y_1, \dots, y_{i-j+1}) = \frac{j!}{i!} B_{i,j}(1!y_1, 2!y_2, \dots, (i-j+1)!y_{i-j+1})$$

and the complete (ordinary) Bell polynomials

$$\hat{G}_i(y_1, \dots, y_i) = G_i(1, \dots, 1; 1!y_1, 2!y_2, \dots, i!y_i).$$

Example 17: To get $\hat{B}_{5,3}(y_1, y_2, y_3)$ run

```
> oBellPol(5, 3)
[1] 1/120( 360(y1)(y2^2) + 360(y1^2)(y3) )
```

To get $\hat{G}_3(y_1, y_2, y_3, y_4)$ run

```
> oBellPol(4)
[1] 1/24( 24(y1^4) + 72(y1^2)(y2) + 24(y2^2) + 48(y1)(y3) + 24(y4) )
```

The e_eBellPol function evaluates the exponential Bell polynomials when the indeterminates are substituted with numerical values. In Table 1 some special sequence of numbers are given obtained using this procedure.

Table 1: Numerical sequences (second column) obtained evaluating the exponential Bell polynomials (last column) when suitable numerical values replace indeterminates.

	Sequence	Bell polynomials
Lah numbers	$\frac{i!}{j!} \binom{i-1}{j-1}$	$B_{i,j}(1!, 2!, 3!, \dots)$
Stirling numbers of first kind	$s(i, j)$	$B_{i,j}(0!, -1!, 2!, \dots)$
unsigned Stirling numbers of first kind	$ s(i, j) $	$B_{i,j}(0!, 1!, 2!, \dots)$
Stirling numbers of second kind	$S(i, j)$	$B_{i,j}(1, 1, 1, \dots)$
idempotent numbers	$\binom{i}{j} j^{i-j}$	$B_{i,j}(1, 2, 3, \dots)$
Bell numbers	B_i	$\sum_{j=0}^i B_{i,j}(1, 1, 1, \dots)$

By default, the e_eBellPol function returns the Stirling numbers of second kind, as the following example shows.

Example 18: To get $S(5,3)$ run

```
> e_eBellPol(5, 3)
[1] 25
> e_eBellPol(5, 3, c(1, 1, 1, 1, 1))
[1] 25
```

To get the 5-th Bell number B_5 run

```
> e_eBellPol(5)
[1] 52
```

To get $s(5,3)$ run

```
> e_eBellPol(5, 3, c(1, -1, 2, -6, 24))
[1] 35
```

6 Composition of formal power series

In (3), suppose f_t the t -th coefficient of $f(x)$ and $g_{1;s_1}, \dots, g_{n;s_n}$ the s_1 -th, \dots , s_n -th coefficients of $g_1(z), \dots, g_n(z)$ respectively. Using multi-index partitions, the multivariate Faà di Bruno's formula (6) can be written as (E. Di Nardo, Guarino, and Senato 2011)

$$h_i = i! \sum_{\substack{\Lambda = (s_1, \dots, s_n) \\ s_1 + \dots + s_n = i}} f_{(l(\Lambda), \dots, l(\tilde{\Lambda}))} \frac{g_{1,\Lambda} \cdots g_{n,\tilde{\Lambda}}}{\Lambda! \cdots \tilde{\Lambda}! m(\Lambda)! \cdots m(\tilde{\Lambda})!} \tag{37}$$

where $g_{1,\Lambda}, \dots, g_{n,\tilde{\Lambda}}$ are given in (29) and the sum is over all the partitions $\Lambda \vdash s_1, \dots, \tilde{\Lambda} \vdash s_n$, with s_1, \dots, s_n m -tuples of non-negative integers such that $s_1 + \dots + s_n = i$.

The MFB function generates all the summands of (37). Its first step is to find the set $\tilde{p}(n, i)$ of all the compositions of i in n parts, that is all the n -tuples (s_1, \dots, s_n) of non-negative integer m -tuples such that $s_1 + \dots + s_n = i$. This task is performed by the mkT function.

Function mkT

- i) Find all the partitions $\Lambda \vdash i$, using the mkmSet function.
- ii) Select the first partition Λ . If $l(\Lambda) = n$, then the columns of Λ are the m -tuples (s_1, \dots, s_n) such that $s_1 + \dots + s_n = i$. If $l(\Lambda) < n$, add $n - l(\Lambda)$ zero columns to Λ .

- iii) Generate all the permutations of the columns of Λ as collected at step ii).
- iv) Repeat steps ii) and iii) for each partition Λ carried out at step i).

In the mkT function an input flag variable permits to obtain the output in a more compact set up. See the following example.

Example 19: Suppose we are looking for the elements of the set $\tilde{p}(2, (2, 1))$, that is the pairs (s_1, s_2) such that $s_1 + s_2 = (2, 1)$. Then run

```
> mkT(c(2, 1), 2, TRUE)
[( 0 1 )( 2 0 )]
[( 2 0 )( 0 1 )]
[( 1 0 )( 1 1 )]
[( 1 1 )( 1 0 )]
[( 2 1 )( 0 0 )]
[( 0 0 )( 2 1 )]
```

Consider the partitions of $(2, 1)$ as given in Example 2. Note that $[(2 1)(0 0)]$ and $[(0 0)(2 1)]$ are obtained adding a zero column to the partition $[(2 1), 1]$, and then permuting the two columns. No zero columns are added to $[(2 0)(0 1)]$ as the length of the partition is 2. The same is true for $[(0 1)(2 0)]$ or $[(1 1)(1 0)]$ which are only permuted.

The MFB function produces the multivariate Faà di Bruno’s formula (37) making use of the following steps.

Function MFB

- i) Find all the m -tuples (s_1, \dots, s_n) in $\tilde{p}(n, i)$ using the mkT function.
- ii) Let y_1, \dots, y_n be indeterminates. For each $j = 1, \dots, n$, compute all the partitions $\Lambda \vdash s_j$ using the mkmSet function and find the explicit expression of the polynomial

$$q_{j,s_j}(y_j) = s_j! \sum_{\Lambda \vdash s_j} y_j^{l(\Lambda)} \frac{\delta_{j,\Lambda}}{\Lambda!m(\Lambda)!}$$

- iii) Make the products $q_{1,s_1}(y_1) \cdots q_{n,s_n}(y_n)$ in the multivariable polynomial

$$h_i(y_1, \dots, y_n) = \sum_{(s_1, \dots, s_n) \in \tilde{p}(n, i)} \binom{i}{s_1, \dots, s_n} q_{1,s_1}(y_1) \cdots q_{n,s_n}(y_n)$$

and compute its explicit expression.

- iv) In the explicit expression of the polynomial $h_i(y_1, \dots, y_n)$ as carried out at the previous step iii), replace the occurrences of the products $y_1^{l(\Lambda)} \cdots y_n^{l(\tilde{\Lambda})}$ with $f_{(l(\Lambda), \dots, l(\tilde{\Lambda}))}$.

Step iii) is performed by the joint function. This function is not directly accessible in the package, as defined locally in the MFB function. The joint function realizes a recursive pair matching: each coefficient $g_{1,\Lambda}$ of $q_{1,s_1}(y_1)$ is matched with each coefficient $g_{2,\tilde{\Lambda}}$ of $q_{2,s_2}(y_2)$, then each paired coefficient $g_{1,\Lambda}g_{2,\tilde{\Lambda}}$ is matched with each coefficient g_{3,Λ^*} of $q_{3,s_3}(y_3)$ and so on. Step iv) consists of multiplying each coefficient found at step iii) with f_t , where t is the multi-index whose j -th component gives how many times g_j appears in this coefficient. See the following example.

Example 20: Suppose $n = m = 2$ and $i = (1, 1)$. To get $h_{(1,1)}$ in (37) run

```
> MFB(c(1, 1), 2)
[1] f[1, 1]g1[0, 1]g2[1, 0] + f[1, 1]g1[1, 0]g2[0, 1] + f[2, 0]g1[0, 1]g1[1, 0] +
f[1, 0]g1[1, 1] + f[0, 2]g2[0, 1]g2[1, 0] + f[0, 1]g2[1, 1]
```

Taking into account (4), in the previous output $f[i, j]$ corresponds to $f_{(i,j)}$ as well as $g1[i, j]$ and $g2[i, j]$ correspond to $g_{1;(i,j)}$ and $g_{2;(i,j)}$ respectively for $i, j = 0, 1, 2$. Note that $g1[1, 1]$ is multiplied with $f[1, 0]$ as there is one occurrence of $g1$ and no occurrence of $g2$. In the same way, $g1[1, 0]g1[0, 1]$ is multiplied with $f[2, 0]$ as there are two occurrences of $g1$ and no occurrence of $g2$ and $g1[1, 0]g2[0, 1]$ is multiplied with $f[1, 1]$ as there is one occurrence of $g1$ and one occurrence of $g2$ and so on. Compare the previous output with the one obtained in Maple running

diff(f(g1(x1,x2),g2(x1,x2),x1,x2):

$$\begin{aligned}
 & D_{2,2}(f)(g1(x1, x2), g2(x1, x2)) \left(\frac{\partial}{\partial x_1} g2(x1, x2)\right) \left(\frac{\partial}{\partial x_2} g2(x1, x2)\right) \\
 & + D_{1,2}(f)(g1(x1, x2), g2(x1, x2)) \left(\frac{\partial}{\partial x_2} g1(x1, x2)\right) \left(\frac{\partial}{\partial x_1} g2(x1, x2)\right) \\
 & + D_{1,1}(f)(g1(x1, x2), g2(x1, x2)) \left(\frac{\partial}{\partial x_1} g1(x1, x2)\right) \left(\frac{\partial}{\partial x_2} g2(x1, x2)\right) \\
 & + D_{1,1}(f)(g1(x1, x2), g2(x1, x2)) \left(\frac{\partial}{\partial x_1} g1(x1, x2)\right) \left(\frac{\partial}{\partial x_2} g1(x1, x2)\right) \\
 & + D_2(f)(g1(x1, x2), g2(x1, x2)) \left(\frac{\partial^2}{\partial x_2 \partial x_1} g2(x1, x2)\right) \\
 & + D_1(f)(g1(x1, x2), g2(x1, x2)) \left(\frac{\partial^2}{\partial x_2 \partial x_1} g1(x1, x2)\right)
 \end{aligned}$$

where $D_1(f)$ denotes $\partial f(x_1, x_2) / \partial x_1$, $D_2(f)$ denotes $\partial f(x_1, x_2) / \partial x_2$ and similarly

$$D_{1,1}(f) \leftarrow \frac{\partial^2 f(x_1, x_2)}{\partial x_1^2}, D_{2,2}(f) \leftarrow \frac{\partial^2 f(x_1, x_2)}{\partial x_2^2}, D_{1,2}(f) \leftarrow \frac{\partial^2 f(x_1, x_2)}{\partial x_1 \partial x_2}.$$

The eMFB function evaluates the multivariate Faà di Bruno’s formula (37) when the coefficients of the formal power series f and g_1, \dots, g_n in (4) are substituted with numerical values.

Example 21: To evaluate the output of Example 20 for some numerical values of the coefficients, run

```

> cfVal<-"f[0,1]=2, f[0,2]=5, f[1,0]=13, f[1,1]=-4, f[2,0]=0"
> cgVal<-"g1[0,1]=-2.1, g1[1,0]=2, g1[1,1]=3.1, g2[0,1]=5, g2[1,0]=0, g2[1,1]=6.1"
> cVal<-paste0(cfVal, " ", cgVal)
> e_MFB(c(1,1), 2, cVal)
[1] 12.5
    
```

The polynomial families discussed in the previous sections are generated using the MFB function. Indeed, the generalized (complete exponential) Bell polynomials in (28) are coefficients of the following formal power series

$$H(y_1, \dots, y_n; z) = 1 + \sum_{|i|>0} h_i(y_1, \dots, y_n) \frac{z^i}{i!} = \exp \left[\sum_{i=1}^n y_i (g_i(z) - 1) \right], \tag{38}$$

which turns to be a composition (3), with $f(x_1, \dots, x_n) = \exp(x_1 y_1 + \dots + x_n y_n)$ and $f_t = y_1^{t_1} \dots y_n^{t_n}$ for $t \in \mathbb{N}_0^n$. In this case, y_1, \dots, y_n play the role of indeterminates. The i -th coefficient $h_i(y_1, \dots, y_n)$ - output of the GCBellPol function - is obtained from the multivariate Faà di Bruno’s formula (37) dealing with y_1, \dots, y_n as they were constants. When $\{g_1(z), \dots, g_n(z)\}$ are the same formal power series $g(z)$, the formal power series $H(y_1, \dots, y_n; z)$ in (38) reduces to

$$H(y_1, \dots, y_n; z) = \exp \left[(y_1 + \dots + y_n)(g(z) - 1) \right] \tag{39}$$

with coefficients $\tilde{h}_i(y_1, \dots, y_n)$ as given in the previous section.

If $n = 1$ then $H(y_1, \dots, y_n; z)$ reduces to the composition $\exp [y(g(z) - 1)]$ whose coefficients are the polynomials given in (30). More in general the coefficients of $f(g(z) - 1)$ are

$$h_i = \sum_{\Lambda \vdash i} d_\Lambda f_{1(\Lambda)} g_\Lambda \tag{40}$$

where the sum is over all the partitions $\Lambda = (\lambda_1^{r_1}, \lambda_2^{r_2}, \dots) \vdash i$, d_Λ is given in (19) and $g_\Lambda = g_{\lambda_1}^{r_1} g_{\lambda_2}^{r_2} \dots$. If also $m = 1$, then h_i in (40) reduces to

$$h_i = \sum_{\lambda \vdash i} d_\lambda f_{1(\lambda)} g_\lambda \tag{41}$$

where the sum is over all the partitions $\lambda = (1^{r_1}, 2^{r_2}, \dots) \vdash i$, d_λ is given in (16) and $g_\lambda = g_1^{r_1} g_2^{r_2} \dots$. Formula (41) corresponds to the univariate Faà di Bruno’s formula and gives the i -th coefficient of $f(g(z) - 1)$ with

$$f(x) = 1 + \sum_{j \geq 1} f_j \frac{x^j}{j!} \quad \text{and} \quad g(z) = 1 + \sum_{s \geq 1} g_s \frac{z^s}{s!}.$$

Example 22: To get h_5 in (41) run

```

> MFB(c(5), 1)
[1] f[5]g[1]^5 + 10f[4]g[1]^3g[2] + 15f[3]g[1]g[2]^2 + 10f[3]g[1]^2g[3] +
10f[2]g[2]g[3] + 5f[2]g[1]g[4] + f[1]g[5]
    
```

For instance, the i -th general partition polynomial in (33) is generated using the MFB function: in such a case the univariate Faà di Bruno's formula (41) is generated with $\{y_s\}$ in place of $\{g_s\}$ and $\{a_j\}$ in place of $\{f_j\}$.

Examples of how to generate polynomials not included in the `kStatistics` package

In the following we give some suggestions on how to use the R codes of the `kStatistics` package to generate additional polynomial families.

The `pPart` function is an example of how to use the univariate Faà di Bruno's formula and a symbolic strategy different from those presented so far. Indeed the `pPart` function generates the so-called partition polynomial $F_i(y)$ of degree i , whose coefficients are the number of partitions of i with j parts for $j = 1, \dots, i$ (Boyer and Goh 2008). The partition polynomial $F_i(y)$ is obtained from the univariate Faà di Bruno's formula (41) setting

$$f_j = 1/i! \quad \text{and} \quad g_s^{r_s} = (s!)^{r_s} r_s! y^{r_s} \tag{42}$$

for $s = 1, \dots, i - j + 1, j = 1, \dots, i$ and $r_s = 1, \dots, i$. Note the symbolic substitution of $g_s^{r_s}$ with the powers y^{r_s} .

Example 23: To get $F_5(y)$ run

```
> pPart(5)
[1] y^5 + y^4 + 2y^3 + 2y^2 + y
```

Note that $F_5(y)$ is obtained from the output of Example 22 using (42).

Example 24: The following code shows how to evaluate $F_{11}(y)$ in $y = 7$.

```
> s<-pPart(11)          # generate the partition polynomial of degree 11
> s<-paste0("1",s)     # add the coefficient to the first term
> s<-gsub(" y", "1y",s) # replace the variable y without coefficient
> s<-gsub("y", "*7",s)  # assign y = 7
> eval(parse(text=s))   # evaluation of the expression
[1] 3.476775e+182
```

We give a further example on how to generate a polynomial family not introduced so far but still coming from (41) for suitable choices of $\{f_j\}$ and $\{g_s\}$. Consider the elementary symmetric polynomials in the indeterminates y_1, \dots, y_n

$$e_i(y_1, \dots, y_n) = \begin{cases} \sum_{1 \leq j_1 < \dots < j_i \leq n} y_{j_1} \cdots y_{j_i}, & 1 \leq i \leq n, \\ 0, & i > n. \end{cases} \tag{43}$$

A well-known result (Charalambides 2002) states that these polynomials can be expressed in terms of the power sum symmetric polynomials (14) in the same indeterminates y_1, \dots, y_n , using the general partition polynomials (34), that is

$$e_i = \frac{(-1)^i}{i!} G_i(1, \dots, 1; -p_1, -1!p_2, -2!p_3, \dots, -(i-1)!p_i) \tag{44}$$

for $i = 1, \dots, n$. The following `e2p` function expresses the i -th elementary symmetric polynomial e_i in terms of the power sum symmetric polynomials p_1, \dots, p_i , using (44) and the MFB function.

```
> e2p <- function(n=0){
+   v<-MFB(n,1);          # Call the MFB Function
+   v<-MFB2Set( v );      # Expression to vector
+   for (j in 1:length(v)) {
+     # ----- read -----[ fix block ]-----#
+     c <- as.character(v[[j]][2]); # coefficient
+     x <- v[[j]][3];          # variable
+     i <- v[[j]][4];          # subscript
+     k <- strtoi(v[[j]][5]);   # power
+     # ----- change -----#
+     if (x=="f") {
+       c<-paste0(c, "*" ( -1)^",n,");
```



```

+         x<-"";
+         i<-"";
+     }
+     else if (x=="g") {
+         c<-paste0(c,"*((-factorial(",strtoi(i)-1,")^",k,")");
+         x<-paste0("p",i,ifelse(k>1,paste0("^",k,""),")");
+         i<-"";k<-1;
+     }
+     # ----- write -----[ fix block ]-----#
+     v[[j]][2] <- c;
+     v[[j]][3] <- x;
+     v[[j]][4] <- i;
+     v[[j]][5] <- k;
+     # -----#
+ }
+ noquote(paste0("1/",factorial(n),"( ",Set2expr(v), " )"));
+ }

```

This function starts by initializing the vector v with (41) by means of the MFB function. There is a first code snippet [fix block] for extracting a set with the coefficients, variables, indexes and powers of v by means of the MFB2Set function. This first code snippet should not be changed whatever polynomial family we are generating. The second code snippet change includes instructions that can be changed according to the expressions of the coefficients $\{f_j\}$ and $\{g_s\}$ in (41). To get (44), we set $f_j = 1$ and $g_s = -(s-1)!p_s$. Once these coefficients have been changed, the last code snippet [fix block] updates the vector v . The Set2expr function assembles the final expression.

Example 25: To get e_4 in (44) run

```

> e2p(4)
[1] 1/24( (p1^4) - 6(p1^2)(p2) + 3(p2^2) + 8(p1)(p3) - 6(p4) )

```

7 Concluding remarks

We have developed the **kStatistics** package with the aim to generate univariate and multivariate k -statistics/polykeys, together with the multivariate Faà di Bruno's formula and various user-friendly functions related to this formula. The paper briefly introduces the combinatorial tools involved in the package and presents, in detail, the core function of the package which generates multi-index partitions. We emphasize that the algorithms presented here have been designed with the aid of the umbral calculus, even if we did not mention this method in the paper.

One of the main applications we have dealt with is the generation and evaluation of various families of polynomials: from generalized complete Bell polynomials to general partition polynomials, from partial Bell polynomials to complete Bell polynomials. Numerical sequences obtained from the Bell polynomials can also be generated.

All these utilities intend to make the **kStatistics** package a useful tool not only for statisticians but also for users who need to work with families of polynomials usually available in symbolic software or tables. Indeed, we have provided examples on how to generate polynomial families not included in the package but which can still be recovered using the Faà di Bruno's formula and suitable strategies, both numerical and symbolic. Following this approach, also the estimations of joint cumulants or products of joint cumulants is one further example of symbolic strategy coming from the multivariate Faà di Bruno's formula.

Future works consist in expanding the **kStatistics** package by including extensions of the multivariate Faà di Bruno's formula, as addressed in Bernardini, Natalini, and Ricci (2005) and references therein, aiming to manage nested compositions, as the BellY function in the Wolfram Language and System does. Moreover, further procedures can be inserted relied on symbolic strategies not apparently related to the multivariate Faà di Bruno's formula but referable to this formula, as for example the central Bell polynomials (Kim, Kim, and Jang 2019).

The results in this paper were obtained using the **kStatistics** 2.1.1 package. The package is currently available with a general public license (GPL) from the Comprehensive R Archive Network.

8 Acknowledgements

The authors would like to thank the reviewers for their constructive feedback.

References

- Arnqvist, Natalya Pya, Vassilly Voinov, Rashid Makarov, and Yevgeniy Voinov. 2021. *Nilde: Nonnegative Integer Solutions of Linear Diophantine Equations with Applications*. <https://CRAN.R-project.org/package=nilde>.
- Bernardini, A., P. Natalini, and P. E. Ricci. 2005. "Multidimensional Bell Polynomials of Higher Order." *Comput. Math. Appl.* 50 (10-12): 1697–1708. <https://doi.org/10.1016/j.camwa.2005.05.008>.
- Boyer, Robert P., and William M. Y. Goh. 2008. "Partition Polynomials: Asymptotics and Zeros." In *Tapas in Experimental Mathematics*, 457:99–111. Contemp. Math. Amer. Math. Soc., Providence, RI. <https://doi.org/10.1090/conm/457/08904>.
- Brown, James Ward. 1979. "On Multivariable Sheffer Sequences." *J. Math. Anal. Appl.* 69 (2): 398–410. [https://doi.org/10.1016/0022-247X\(79\)90151-3](https://doi.org/10.1016/0022-247X(79)90151-3).
- Chacón, José E., and Tarn Duong. 2015. "Efficient Recursive Algorithms for Functionals Based on Higher Order Derivatives of the Multivariate Gaussian Density." *Statistics and Computing* 25 (5): 959–74. <https://doi.org/10.1007/s11222-014-9465-1>.
- Charalambides, Charalambos A. 2002. *Enumerative Combinatorics*. CRC Press Series on Discrete Mathematics and Its Applications. Chapman & Hall/CRC, Boca Raton, FL.
- Clausen, Andrew, and Serguei Sokol. 2020. *Deriv: R-Based Symbolic Differentiation*. <https://CRAN.R-project.org/package=Deriv>.
- Constantine, G. M., and T. H. Savits. 1996. "A Multivariate Faà Di Bruno Formula with Applications." *Transactions of the American Mathematical Society* 348 (2): 503–20. <https://doi.org/10.1090/S0002-9947-96-01501-2>.
- De Leeuw, J. 2012. "Multivariate Cumulates in R." <https://escholarship.org/uc/item/1fw1h53c>.
- Di Nardo, E. 2016a. "On Multivariable Cumulant Polynomial Sequences with Applications." *Journal of Algebraic Statistics* 7 (1): 72–89. <https://doi.org/10.18409/jas.v7i1.49>.
- . 2016b. "On Photon Statistics Parametrized by a Non-Central Wishart Random Matrix." *Journal of Statistical Planning and Inference* 169: 1–12. <https://doi.org/10.1016/j.jspi.2015.07.002>.
- Di Nardo, E., and G. Guarino. 2019. *Unbiased Estimators for Cumulant Products*. <https://cran.r-project.org/web/packages/kStatistics/index.html>.
- . 2021. *kStatistics: Unbiased Estimators for Cumulant Products and Faà Di Bruno's Formula*. <https://CRAN.R-project.org/package=kStatistics>.
- Di Nardo, E., G. Guarino, and D. Senato. 2011. "A New Algorithm for Computing the Multivariate Faà Di Bruno's Formula" 217 (13): 6286–95. <https://doi.org/10.1016/j.amc.2011.01.001>.
- Di Nardo, Elvira. 2011. "Symbolic Calculus in Mathematical Statistics: A Review." *Séminaire Lotharingien de Combinatoire* 67: Art. B67a, 72. <https://www.mat.univie.ac.at/~slc/wpapers/s67dinardo.pdf>.
- Di Nardo, E., M. Marena, and P. Semeraro. 2020. "On Non-Linear Dependence of Multivariate Subordinated Lévy Processes." *Statistics & Probability Letters* 166: 108870–77. <https://doi.org/10.1016/j.spl.2020.108870>.
- Di Nardo, E., and I. Oliva. 2011. "On a Symbolic Version of Multivariate Lévy Processes." *American Institute of Physics Conference Proceedings* 1389 (1): 345–48. <https://doi.org/10.1063/1.3636735>.
- Dimitrakopoulos, Roussos, Hussein Mustapha, and Erwan Gloaguen. 2010. "High-Order Statistics of Spatial Random Fields: Exploring Spatial Cumulants for Modeling Complex Non-Gaussian and Non-Linear Phenomena." *Mathematical Geosciences* 42 (1): 65–99. <https://doi.org/10.1007/s11004-009-9258-9>.
- Domino, Krzysztof, Piotr Gawron, and Łukasz Paweła. 2018. "Efficient Computation of Higher-Order Cumulant Tensors." *SIAM J. Sci. Comput.* 40 (3): A1590–610. <https://doi.org/10.1137/17M1149365>.
- Ferreira, Pedro G, Joao Magueijo, and Joseph Silk. 1997. "Cumulants as Non-Gaussian Qualifiers." *Physical Review D* 56 (8): 4592. <https://doi.org/10.1103/PhysRevD.56.4592>.
- Geng, Min, Huaqing Liang, and Junwu Wang. 2011. "Research on Methods of Higher-Order Statistics for Phase Difference Detection and Frequency Estimation." In *2011 4th International Congress on Image and Signal Processing*, 4:2189–93. <https://doi.org/10.1109/CISP.2011.6100593>.
- Giannakis, G. B. 1987. "Cumulants: A Powerful Tool in Signal Processing." *Proceedings of the IEEE* 75 (9): 1333–34. <https://doi.org/10.1109/PROC.1987.13884>.
- Guarino, G., D. Senato, and E. Di Nardo. 2009. "Fast Maple Algorithms for k -Statistics, Polykays and Their Multivariate Generalization." <https://www.maplesoft.com/applications/view.aspx?SID=33041>.
- Hankin, R. K. S. 2006. "Additive Integer Partitions in r ." *Journal of Statistical Software, Code Snippets* 16.
- Hardy, Michael. 2006. "Combinatorics of Partial Derivatives." *Electronic Journal of Combinatorics* 13 (1): Research Paper 1, 13. http://www.combinatorics.org/Volume_13/Abstracts/v13i1r1.html.
- Hernández Encinas, L., and J. Muñoz Masqué. 2003. "A Short Proof of the Generalized Faà Di Bruno's Formula." *Applied Mathematics Letters. An International Journal of Rapid Publication* 16 (6): 975–79. [https://doi.org/10.1016/S0893-9659\(03\)90026-7](https://doi.org/10.1016/S0893-9659(03)90026-7).

- Jammalamadaka, S. Rao, T. Subba Rao, and György Terdik. 2006. "Higher Order Cumulants of Random Vectors and Applications to Statistical Inference and Time Series." *Sankhyā. The Indian Journal of Statistics* 68 (2): 326–56. <https://www.jstor.org/stable/25053499>.
- Joanes, Derrick N, and Christine A Gill. 1998. "Comparing Measures of Sample Skewness and Kurtosis." *Journal of the Royal Statistical Society: Series D (The Statistician)* 47 (1): 183–89.
- Kim, Taekyun, Dae San Kim, and Gwan-Woo Jang. 2019. "On Central Complete and Incomplete Bell Polynomials i." *Symmetry* 11 (2). <https://www.mdpi.com/2073-8994/11/2/288>.
- Leipnik, Roy B., and Charles E. M. Pearce. 2007. "The Multivariate Faà Di Bruno Formula and Multivariate Taylor Expansions with Explicit Integral Remainder Term." *The ANZIAM Journal. The Australian & New Zealand Industrial and Applied Mathematics Journal* 48 (3): 327–41. <https://doi.org/10.1017/S1446181100003527>.
- Ma, Tsoy-Wo. 2009. "Higher Chain Formula Proved by Combinatorics." *Electronic Journal of Combinatorics* 16 (1): N21, 7. <https://doi.org/10.37236/259>.
- McCullagh, Peter. 1987. *Tensor Methods in Statistics*. Monographs on Statistics and Applied Probability. Chapman & Hall, London.
- Mihoubi, Miloud. 2008. "Polynômes Multivariés de Bell Et Polynômes de Type Binomial." PhD thesis, L'Université des Sciences et de la Technologie Houari Boumedienne.
- Mishkov, Rumen L. 2000. "Generalization of the Formula of Faà Di Bruno for a Composite Function with a Vector Argument." *International Journal of Mathematics and Mathematical Sciences* 24 (7): 481–91. <https://doi.org/10.1155/S0161171200002970>.
- Nguwi, Jiang Yu, Guillaume Penent, and Nicolas Privault. 2022. "A Deep Branching Solver for Fully Nonlinear Partial Differential Equations." arXiv. <https://arxiv.org/abs/2203.03234>.
- Oualla, Hicham, Rachid Fateh, Anouar Darif, Said Safi, Mathieu Pouliquen, and Miloud Frikel. 2021. "Channel Identification Based on Cumulants, Binary Measurements, and Kernels." *Systems* 9 (2). <https://doi.org/10.3390/systems9020046>.
- Peccati, Giovanni, and Murad S. Taqqu. 2011. "Combinatorial Expressions of Cumulants and Moments." In *Wiener Chaos: Moments, Cumulants and Diagrams.*, edited by Milano Springer, 1:201–13. Bocconi & Springer Series.
- Privault, Nicolas. 2021. "Recursive Computation of the Hawkes Cumulants." *Statistics and Probability Letters*, 109161. <https://doi.org/10.1016/j.spl.2021.109161>.
- Rao, T. Subba, and W. K. Wong. 1999. "Some Contributions to Multivariate Nonlinear Time Series and to Bilinear Models." In *Asymptotics, Nonparametrics, and Time Series*, 158:259–94. Statist. Textbooks Monogr. Dekker, New York.
- Reiner, David L. 1976. "Multivariate Sequences of Binomial Type." *Studies in Applied Mathematics* 57 (2): 119–33. <https://doi.org/10.1002/sapm1977572119>.
- Robson, D. S. 1957. "Applications of Multivariate Polykays to the Theory of Unbiased Ratio-Type Estimation." *Journal of the American Statistical Association* 52 (280): 511–22. <https://www.tandfonline.com/doi/abs/10.1080/01621459.1957.10501407>.
- Roman, Steven. 1984. *The Umbral Calculus*. Vol. 111. Pure and Applied Mathematics. Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], New York.
- Rose, C., and M. D. Smith. 2002. *Mathematical Statistics with Mathematica®*. Springer Texts in Statistics. Springer-Verlag, New York. <https://doi.org/10.1007/978-1-4612-2072-5>.
- Savits, Thomas H. 2006. "Some Statistical Applications of Faà Di Bruno." *Journal of Multivariate Analysis* 97 (10): 2131–40. <https://doi.org/10.1016/j.jmva.2006.03.001>.
- Shabat, A. B., and M. Kh. Efendiev. 2017. "On Applications of Faà-Di-Bruno Formula." *Ufa Mathematical Journal* 9 (3): 131–36. <https://doi.org/10.1007/s12572-017-0181-x>.
- Shrivastava, HSP. 2002. "Multiindex Multivariable Hermite Polynomials." *Math. Comput. Appl.* 7 (2): 139–49. <https://doi.org/10.3390/mca7020139>.
- Smith, Kevin D. 2020. "A Tutorial on Multivariate k -Statistics and Their Computation." arXiv. <https://arxiv.org/abs/2005.08373>.
- Smith, Kevin D., Saber Jafarpour, Ananthram Swami, and Francesco Bullo. 2022. "Topology Inference with Multivariate Cumulants: The Möbius Inference Algorithm." *IEEE/ACM Transactions on Networking*, 1–15. <https://doi.org/10.1109/TNET.2022.3164336>.
- Staudte, Benjamin, Stefan Rotter, and Sonja Grün. 2010. "CuBIC: Cumulant Based Inference of Higher-Order Correlations in Massively Parallel Spike Trains." *Journal of Computational Neuroscience* 29 (1-2): 327–50. <https://doi.org/10.1007/s10827-009-0195-x>.
- Stuart, Alan, and J. Keith Ord. 1994. *Kendall's Advanced Theory of Statistics. Vol. 1*. Sixth. Edward Arnold, London; copublished in the Americas by Halsted Press [John Wiley & Sons, Inc.], New York.
- Voinov, Vassily, and Natalya Pya Arnvist. 2017. "R-Software for Additive Partitioning of Positive Integers." *Mathematical Journal* 17 (1): 69–76. <http://www.math.kz/media/journal/journal2018-05-1574083.pdf>.
- Withers, Christopher S., and Saralees Nadarajah. 2010. "Multivariate Bell Polynomials." *International Journal of Computer Mathematics* 87 (11): 2607–11. <https://doi.org/10.1080/00207160802702418>.

Elvira Di Nardo
University of Turin
Department of Mathematics "G.Peano"
Via Carlo Alberto 10, 10123 Turin (Italy)
<https://www.elviradinardo.it>
ORCID: 0000-0003-3447-9155
elvira.dinardo@unito.it

Giuseppe Guarino
Università Cattolica del Sacro Cuore
Largo Agostino Gemelli 8, 00168, Rome (Italy)
giuseppe.guarino@rete.basilicata.it

iccCounts: An R Package to Estimate the Intraclass Correlation Coefficient for Assessing Agreement with Count Data

by Josep L. Carrasco

Abstract The intraclass correlation coefficient (ICC) is a widely used index to assess agreement with continuous data. The common approach for estimating the ICC involves estimating the variance components of a linear mixed model under assumptions such as linearity and normality of effects. However, if the outcomes are counts these assumptions are not met and the ICC estimates are biased and inefficient. In this situation, it is necessary to use alternative approaches that are better suited for count data. Here, the `iccCounts` R package is introduced for estimating the ICC under the Poisson, Negative Binomial, Zero-Inflated Poisson and Zero-Inflated Negative Binomial distributions. The utility of the `iccCounts` package is illustrated by three examples that involve the assessment of repeatability and concordance with count data.

1 Introduction

Repeated measurements are often collected and hierarchically structured in clusters (commonly, in subjects). These repeated measurements can be interchangeable within subjects (i.e. they are replicates). This case is often known as the evaluation of repeatability (Nakagawa and Schielzeth 2010) or intrarater reliability (DeVet et al. 2011). Moreover, the repeated measurements may be structured (not interchangeable) because they were obtained under different experimental conditions, involving different methods or observers. In this case the analysis of agreement is often known as concordance analysis, method comparison analysis (Choudhary and Nagaraja 2017) or inter-rater reliability (DeVet et al. 2011).

Whatever the structure of the repeated measurements, the intraclass correlation coefficient (ICC) is a common index used to assess agreement with continuous data (Fleiss 1986; Carrasco and Jover 2003). The general definition of the ICC is the ratio of the between-clusters variance to the total variance. However, the appropriate ICC has to be defined to afford the different variance components that are involved in the total variance besides the between-clusters variance. These variance components are typically estimated by means of a linear mixed model with common assumptions: that there is linearity between the outcome expectation and the effects (cluster, method, ...); and that the random effects and the random error follow normal distributions. Currently, there are several R packages that estimate the ICC under the Normality assumption for assessing repeatability or concordance (Wolak, Fairbairn, and Paulsen 2012; Carrasco et al. 2013; Stoffel, Nakagawa, and Schielzeth 2017).

However, if the outcomes are counts, such assumptions are not met and the ICC estimates are biased and inefficient (Carrasco and Jover 2005). In this situation, it is necessary to use alternative approaches that are better suited to the properties of count data. The methodology for estimating the ICC for non-normal distributions using generalized linear mixed models (GLMM), and in particular for count data, was developed in Carrasco (2010). The ICC is therefore estimated by the variance components from the appropriate GLMM. The cluster random effect is still distributed as a Normal distribution but the within-cluster variability is assumed to follow a probability distribution function for counts. Stoffel, Nakagawa, and Schielzeth (2017) introduced the `rptR` package which can be used to estimate the ICC assuming a Poisson model for the within-subjects variability.

In the `iccCounts` package introduced here, besides the Poisson distribution, other models as the Negative Binomial, the Zero-Inflated Poisson and the Zero-Inflated Negative Binomial are also considered. These models are useful when overdispersion arises in the Poisson model. Overdispersion means that the variability assumed by the model is lower than that from the data. Therefore, the within-subjects variability and, by extension, the total variance are underestimated and the ICC and its standard error are biased. Thus, the validity of the ICC estimate is closely linked to the validity of the model, so that a goodness-of-fit (GOF) analysis of the model must be performed.

The article is structured as follows: the *Methodology background* section introduces the definition of the ICC, their expressions depending on the model GLLM chosen, some inferential aspects and the validation approach of the GLMM; the issues of the package are described in *Package description* section; in *Examples* section three examples are introduced. Two of them are cases of the repeatability setting whereas the remaining one shows the case of a concordance setting. Finally, the main contributions are summarized in the *Conclusion* section.

2 Methodology background

Experimental design

As mentioned in the introduction, the experimental design depends on the aim of the study: concordance or repeatability. In the case of a concordance study, a sample of n subjects are measured m times by k methods. In this setting, the aim is to analyse the degree of concordance of the measurement methods when assessing the subjects. Note that within-subjects repeated measurements are not interchangeable because they belong to one specific method. It is worthy to noting that the term “methods” is the conventional way to describe the experimental condition of the repeated measurements across subjects in this context, but it might be referred to differently name depending on the context. In this setting, Y_{ijk} stands for the k -th reading made by the j -th method on the i -th subject, with $i = 1, \dots, n$, $j = 1, \dots, m$ and $k = 1, \dots, s$. Hence, there will be three variance components to consider when assessing agreement among the repeated measurements: between-subjects, between-methods and random error variabilities.

In a repeatability study, a sample of n subjects are measured m times. In this case the repeated measurements share the same experimental condition across subjects, therefore they can be considered as interchangeable. Thus, in this setting Y_{ik} stands for the k -th reading made on the i -th subject, with $i = 1, \dots, n$, and $k = 1, \dots, s$. In this case, only two variance components are involved in the evaluation of the agreement: between-subjects and random error variabilities.

Generalized linear mixed model

The estimation of the variance components is carried out by means of generalized linear mixed models (GLMM). The GLMM for the concordance setting (considering subjects and methods effects) is defined as follows:

- Let α_i and β_j be the subjects and methods random effects respectively, with $i = 1, \dots, n$, $j = 1, \dots, m$, that follow Normal distributions with mean 0 and variance σ_α^2 and σ_β^2 . Although the method effect could be a fixed effect by design, when defining the agreement index it is convenient to consider it as a random effect to account for the systematic differences between observers as a source of disagreement (Fleiss 1986; Carrasco and Jover 2003).
- The conditional distribution of Y_{ijk} given α_i and β_j , $f(Y_{ijk}|\alpha_i, \beta_j)$, is a probability density function from the exponential family.
- The conditional mean of Y_{ijk} given α_i and β_j is

$$\mu_{ij} = E(Y_{ijk}|\alpha_i, \beta_j) = g^{-1}(\lambda_i + \alpha_i + \beta_j). \quad (1)$$

where g is called the link function. Here, λ_i is the linear combination of the mean modifying covariates for the i -th subject. Furthermore, the conditional variance of Y_{ijk} given α_i and β_j is defined as $Var(Y_{ijk}|\alpha_i, \beta_j) = \phi h(\mu_{ij})$ where ϕ is the dispersion parameter and $h(\cdot)$ is variance function.

If the methods effect is removed, the GLMM for the repeatability setting is obtained. Thus, depending on the nature of the data the appropriate conditional probability model and link function must be chosen. When analysing count data, the logarithm is commonly used as a link function and models such as Poisson or Negative Binomial are considered.

Intraclass correlation coefficient for count data

The intraclass correlation coefficient (ICC) is calculated as:

$$ICC = \frac{Cov(Y_{ijk}, Y_{ij'k'})}{Var(Y_{ijk})}. \quad (2)$$

where the $Cov(Y_{ijk}, Y_{ij'k'})$ is the marginal (over subjects and observers) covariance between any pair of data from the same subject, whereas $Var(Y_{ijk})$ is the marginal variance of data.

Furthermore, the marginal variance and covariance can be developed as functions of the GLMM parameters (Carrasco 2010):

$$ICC = \frac{Cov(\mu_{ij}, \mu_{ij'})}{Var(\mu_{ij}) + E(\phi h(\mu_{ij}))} \tag{3}$$

This result allows the ICC to be generalized to any distribution fitted with a GLMM.

Carrasco (2010) developed the ICC for Poisson and Negative Binomial distributions, the latter with variance increasing quadratically with the mean (NegBin2), $Var(Y_{ijk}|\alpha_i, \beta_j) = \mu_{ij}(1 + r\mu_{ij})$ (Table 1). A Negative Binomial model with variance increasing linearly with the mean is also considered (NegBin1) (Brooks et al. 2017; Hardin and Hilbe 2007), $Var(Y_{ijk}|\alpha_i, \beta_j) = \mu_{ij}(1 + r)$. It is worth to noting that NegBin1 does not belong to the exponential family (Hardin and Hilbe 2007), therefore this model would not be a proper GLMM. However, it can still be useful to model count data that show overdispersion in a Poisson model.

Additionally, it is possible to define the ICC for the cases of zero inflated models (Table 1). Let's define B_{ijk} as a Bernoulli variable that takes a value of 1 if the reading k on subject i and method j is a structural zero with probability π and 0 otherwise. The observed data, X_{ijk} is the result of $X_{ijk} = Y_{ijk}(1 - B_{ijk})$, where Y_{ijk} is the count variable as defined before. The marginal covariance and variance of X_{ijk} are:

$$Cov(X_{ijk}, X_{ij'k'}) = Cov(Y_{ijk}, Y_{ij'k'}) (1 - \pi)^2 \tag{4}$$

$$Var(X_{ijk}) = Var(Y_{ijk})(1 - \pi) + E^2(Y_{ijk})\pi(1 - \pi) \tag{5}$$

and the general expression of the ICC for zero-inflated data becomes:

$$ICC = \frac{Cov(Y_{ijk}, Y_{ij'k'}) (1 - \pi)}{Var(Y_{ijk}) + E^2(Y_{ijk})\pi} \tag{6}$$

where π stands for the probability of excess of zeros.

Notice that ICCs appearing in Table 1 are for the concordance setting where σ_β^2 stands for the variability between methods. The ICCs for the repeatability setting are obtained just removing σ_β^2 from the equations, i.e. by setting $\sigma_\beta^2 = 0$.

Model	ICC	θ	Model	ICC	θ
Poisson	$\frac{\mu(e^{\sigma_\alpha^2} - 1)}{\mu(e^{\sigma_\alpha^2 + \sigma_\beta^2} - 1) + 1}$	$(\mu, \sigma_\alpha^2, \sigma_\beta^2)$	ZIP	$\frac{\mu(e^{\sigma_\alpha^2} - 1)(1 - \pi)}{\mu(e^{\sigma_\alpha^2 + \sigma_\beta^2} - 1) + 1 + \mu\pi}$	$(\mu, \sigma_\alpha^2, \sigma_\beta^2)$
NegBin1	$\frac{\mu(e^{\sigma_\alpha^2} - 1)}{\mu(e^{\sigma_\alpha^2 + \sigma_\beta^2} - 1) + r + 1}$	$(\mu, \sigma_\alpha^2, \sigma_\beta^2, r)$	ZI-NegBin1	$\frac{\mu(e^{\sigma_\alpha^2} - 1)(1 - \pi)}{\mu(e^{\sigma_\alpha^2 + \sigma_\beta^2} - 1) + 1 + r + \mu\pi}$	$(\mu, \sigma_\alpha^2, \sigma_\beta^2, r, \pi)$
NegBin2	$\frac{\mu(e^{\sigma_\alpha^2} - 1)}{\mu((r+1)e^{\sigma_\alpha^2 + \sigma_\beta^2} - 1) + 1}$	$(\mu, \sigma_\alpha^2, \sigma_\beta^2, r)$	ZI-NegBin2	$\frac{\mu(e^{\sigma_\alpha^2} - 1)(1 - \pi)}{\mu((r+1)e^{\sigma_\alpha^2 + \sigma_\beta^2} - 1) + 1 + \mu\pi}$	$(\mu, \sigma_\alpha^2, \sigma_\beta^2, r, \pi)$

Table 1: ICC expressions. NegBin1 and NegBin2 are the Negative Binomial models with variance increasing linearly and quadratically with the mean respectively; ZI are the zero-inflated models. θ stand for parameters involved in ICC: μ stands for the marginal mean; σ_α^2 is the between-subjects variance; σ_β^2 is the between-methods variability; r is the Negative Binomial's dispersion parameter; and π is the probability of excess of zeros.

Estimation of ICC

The estimation of the ICC involves estimating the GLMM parameters. However, maximum likelihood approach is not straightforward because there is no closed analytical expression for the marginal likelihood besides the Normal case (linear mixed model). Thus, it is necessary to apply numerical methods to approximate the marginal likelihood and to obtain maximum likelihood estimates (Bolker et al. 2009). With regards to the standard error of the ICC, let θ be the GLMM parameters involved in the ICC expression (see Table 1) and Σ the variance-covariance matrix of θ . The asymptotic standard error can be estimated by applying the delta method (Hoef 2012):

$$\text{Var}(ICC) \approx \Delta' \Sigma \Delta \quad (7)$$

where Δ stand for the vector containing the derivatives of ICC respect to θ . Confidence intervals for the ICC are based on asymptotic Normal distribution using the inverse hyperbolic tangent function or Fisher's Z-transformation (Carrasco and Jover 2003).

Validation

The goodness-of-fit (GOF) analysis can be carried out by the computation of randomized quantile residuals (RQR) (Dunn and Smyth 1996; C. Feng, Li, and Sadeghpour 2020). Briefly, the GOF analysis involve the comparison of the RQR from the original data to those obtained by simulation under the fitted model. Simulations of counts based on the fitted model are generated and the model is refitted to each simulated dataset. Using the simulated RQR, envelopes are built as the appropriate quantiles (in relation to the level of significance) of RQR from the refitted models. If the model fits correctly the data it is expected that the original RQR will completely lie within the simulated envelopes.

Additionally, dispersion as well as zero-inflation can be checked by comparing the dispersion and proportion of zeros from the simulated data to those from the original data. Thus, tests for dispersion and zero inflation are carried out by comparing the RQR dispersion and the number of zeros from the original model and data to those from the refitted models and simulated data.

3 Package description

The main function in the `iccCounts` package is `icc_counts` which estimates the ICC under different models for count data. The argument `data` identifies the data set to be analysed. This data set has to be a `data.frame` object with at least two columns: outcome and subject identifier (arguments `y` and `id` respectively).

In the case of estimating the ICC for the concordance setting, a third column with the method identifier must be provided (the argument `met`). The argument `type` is used to identify the setting in which the ICC should be estimated. Valid values are: `rep` (default) for the repeatability setting; and `con` for the concordance setting. The repeatability setting requires that repeated measurements are interchangeable within subjects. This means the experimental conditions of the measurements are the same (replicates), and they come from the same probability distribution function (conditioned to subjects). On the other hand, in the concordance setting the repeated measurements are not interchangeable because their experimental conditions are different, and therefore their probability distribution function, conditioned to subjects, is different (commonly in the mean).

The argument `fam` is used to identify the within-subjects probability model. Valid options are: `poisson` (default) for Poisson model; `nbinom1` and `nbinom2` for Negative Binomial model with variance increasing linearly and quadratically with the mean respectively; `zip` for zero-inflated Poisson model; `zinb1` and `zinb2` for zero-inflated Negative Binomial model with variance increasing linearly and quadratically with the mean.

Once the appropriate setting and model have been chosen, the GLMM is estimated by maximum likelihood via Laplace approximation using the `glmmTMB` package (Brooks et al. 2017). The output of the `icc_counts` function is an object of class `iccc` which is a list with the following components: `model` which contains the generalized linear mixed model estimates; `ICC` which includes the ICC estimate and its standard error and confidence interval; and `varcomp` with the variance components and parameters related to the ICC. Finally, the function `GOF_check` runs the goodness of fit (GOF) analysis of the GLMM fitted to data. This function has three arguments: `x` to denote the `iccc` object to apply the GOF analysis; the `nsim` argument that stands for the number of simulations to run which default value is set to 100; and the `alpha` argument to set the level of significance.

The output of `GOF_check` is an object of class `GOF` which is a list with the following components: `plot_env`, a plot of RQR envelopes with the original RQR; `plot_var`, a plot of the simulated RQR dispersion; `plot_zi`, a plot of the count of zeros in the simulated datasets; `res_var`, the dispersion of RQR from the original sample; `pval_var`, the proportion of simulated RQR dispersion that are greater than the original dispersion that can be interpreted as a simulated P-value to check the goodness of fit on dispersion; `zero_count`, the count of zeros in the original sample; and `pval_zi`, the proportion of simulated zero count that are greater than that of the original sample. It can be interpreted as a simulated P-value to check the hypothesis of zero-inflation. The plots in the list are objects of class `ggplot`, hence users may change the plot themes or add modifications to the components.

Additionally, to describe the differences among the repeated measurements from the same subjects, the function `plot_BA` draws the Bland-Altman plot (Bland and Altman 1995). The difference between

each pair of data from the same subject is represented on the y-axis. The mean of data from the same subject is represented on the x-axis. Additionally, a bar plot with the proportions of differences can be drawn. This plot is a useful way to describe the differences when the range of observed values is small relative to the number of observations (Smith, Ma, and Stafford 2010).

The arguments of `plot_BA` function are: `data`, a data frame containing at least the columns of the outcome and subject's identifier; `y`, a character string indicating the name of the outcome column in the data set; `id` a character string indicating the name of the subjects column in the data set; `rm`, a character string indicating the name of the column that stands for the repeated measurements in the data set. This argument is only needed to identify the differences; `type`, argument used to choose the plot to be drawn. Valid values are: `BA` (default) for the Bland-Altman plot; and `bars` for the bar plot of the differences. Besides the plots, the function provides a dataframe object that contains the data used to generate the plot.

4 Examples

The package includes three real data sets as examples that covers the repeatability and concordance settings.

Sparrow fledglings paternity example

In the Sparrow fledglings paternity example, the incidence of extra-pair paternity (EPP) was monitored over 3 breeding seasons in a sparrow colony in Lundy, an island off the southwest coast of England (Schroeder et al. 2012). One of the aims of the study was to assess the repeatability of counts of fledglings that a male had in every breeding season. Thus, the repeated measurements are assumed to be exchangeable replicates. However, the means of the Social variable by year seem to differ:

```
library(iccCounts)
library(dplyr)
EPP %>% group_by(Year) %>% summarize(Mean=mean(Social),SD=sd(Social))

#> # A tibble: 3 x 3
#>   Year Mean SD
#>   <int> <dbl> <dbl>
#> 1  2003  3.19  3.10
#> 2  2004  2.53  2.21
#> 3  2005  4.5   2.79
```

In case these means were significantly different the repeated measurements could not be considered as exchangeable, and consequently the differences among the means of the repeated measurements should be included in the agreement index. This led us to the concordance setting considering Year as the methods effect.

The first model to consider is the Poisson model. The default option in the `icc_counts` function is the Poisson model, and so it is necessary to specify the name of the data set, the count variable (Social), the subjects identifier (id), the methods variable (Year), and the concordance setting (`type="con"`).

```
EPP_P<-icc_counts(EPP,y="Social",id="id",met="Year",type="con")
ICC(EPP_P)

#>           ICC      SE ICC 95% CI LL 95% CI UL
#> [1,] 0.5284404 0.0866857 0.3383706 0.6770822

VarComp(EPP_P)

#>      mu      BSVar      BMVar
#> 3.172783 0.4002965 0.0798841
```

The function `ICC` applied to the `icc` object shows that the ICC estimate is 0.53 (95% confidence interval: 0.34 - 0.68). Moreover, the function `VarComp` gives the parameters involved in the ICC estimator, which in this case are the overall mean, the between-subjects variance and the between-methods variability (the two latter in log-scale).

However, as mentioned in the previous section, the validity of the ICC estimate is linked to the validity of the model. The function `GOF_check` is applied to the `icc` object to run the simulations and to compute the RQR. The random seed is set for the sake of the reproducibility of the example.

```
set.seed(100)
EPP_P.gof <- GOF_check(EPP_P)

plot(EPP_P.gof)
```

Figure 1a shows the plot of RQR with envelopes generated by simulation. Points on the plot stand for the RQR from the original sample. Notice that a substantial number of points lie outside the envelopes, indicating the fit of the model is unsuitable. The next plot (Figure 1c) shows the density of the RQR variances computed in the simulated samples. The RQR variance from the initial sample is 2.15 (shown inside the square) which is extreme compared to those from the simulations. Indeed, the proportion of simulated variances that are higher than that from the initial sample can be interpreted as a p-value generated by Monte Carlo simulation. This p-value is shown by applying the function *DispersionTest* to the *GOF* object.

```
DispersionTest(EPP_P.gof)

#>      S    P_value
#> 2.077549 0.00990099
```

Additionally, the Social variable has a considerable proportion of zero values (26.4%), and so the excess of zeros could be the cause of the unsuitable fitting of the Poisson model. To check this hypothesis, the third plot generated shows the proportion of zeros in the simulated data sets (Figure 1e). The count of zeros in the sample is 51, which exceeds the expected count under the Poisson model. Again, the proportion of simulated zero counts that are higher than that from the initial sample can be interpreted as a p-value generated by Monte Carlo simulation. This p-value is obtained by applying the function *ZeroTest* to the *GOF* object.

```
ZeroTest(EPP_P.gof)

#> Count    P_value
#>    51 0.00990099
```

Thus, it is necessary to use a model able to provide a proportion of zeros higher than that expected under the Poisson assumption. This model could be the Zero-Inflated Poisson (ZIP) model.

```
EPP_ZIP<-icc_counts(EPP,y="Social",id="id",met="Year",type="con",fam="zip")
ICC(EPP_ZIP)

#>      ICC      SE ICC 95% CI LL 95% CI UL
#> [1,] 0.0477628 0.0362002 -0.02331 0.1183553

VarComp(EPP_ZIP)

#>      mu    BSVar    BMVar      Pi
#> 4.487117 0.033328 0.0328427 0.2446678
```

In this case, the ICC is much lower than in the Poisson model (0.05, 95% CI: -0.02, 0.12) indicating a non-significant ICC. The ICC components are the same as those in the Poisson case (with different values) plus the proportion of excess of zeros (0.24). Next step is to check whether the model correctly fits the data.

```
set.seed(100)
EPP_ZIP.gof <- GOF_check(EPP_ZIP)

plot(EPP_ZIP.gof)
```

Figure 1b shows the model to be appropriate because all the RQR are within the envelopes. Additionally, the dispersion and the proportion of zeros from the initial sample are within the values expected under the ZIP model (Figures 1d and 1f). This fact can be also verified by verifying that dispersion and excess of zeros tests are non significant.

```
DispersionTest(EPP_ZIP.gof)
```

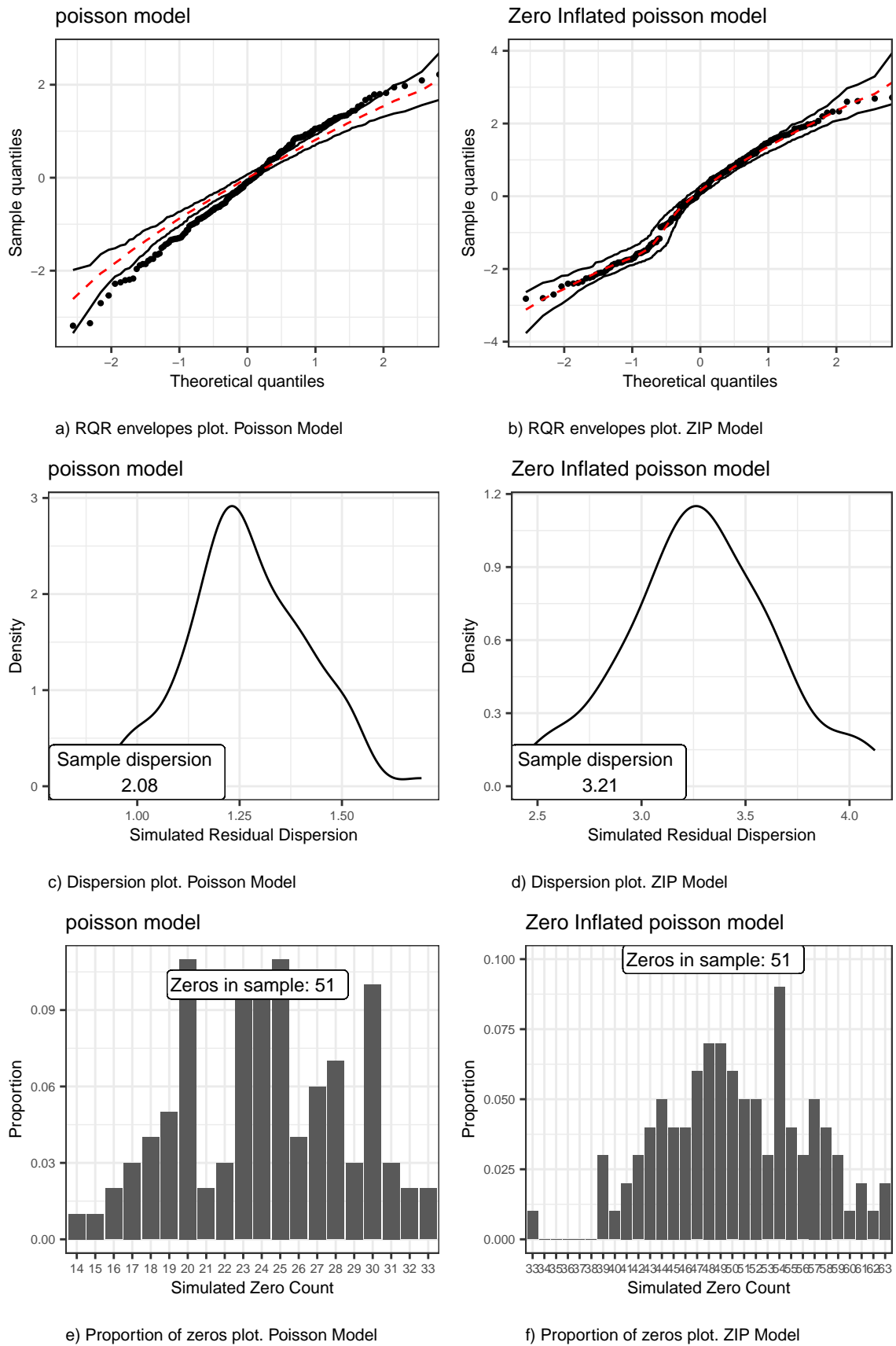


Figure 1: Goodness of fit for Sparrow fledglings paternity example. The Randomized Quantile Residuals (RQR) and counts of zeros of original data are compared to those from simulated data under the fitted model. The plots shown are RQR with envelopes, dispersion of RQR and count of zeros. Left column shows results for Poisson model while the plots for Zero Inflated Poisson (ZIP) model are on right column.

```
#>      S   P_value
#> 3.214152 0.5643564
```

```
ZeroTest(EPP_ZIP.gof)
```

```
#> Count   P_value
#>    51 0.4752475
```

Thus, the ZIP model fits the data appropriately. The next step is to check if the differences in means between years are significant and the concordance setting is therefore justified. With this aim let us apply the function `icc_counts` to the ZIP model but in the repeatability setting.

```
EPP_ZIP_0<-icc_counts(EPP,y="Social",id="id",fam="zip")
```

The component `model` in the `iccc` object is an object of class `glmmTMB` that contains the generalized linear mixed model fit. The `anova` method applied to the model objects gives a comparison of deviances and a likelihood ratio test:

```
anova(EPP_ZIP$model,EPP_ZIP_0$model)
```

```
#> Data: data
#> Models:
#> EPP_ZIP_0$model: y ~ (1 | id), zi=~1, disp=~1
#> EPP_ZIP$model: y ~ met + (1 | id), zi=~1, disp=~1
#>      Df    AIC    BIC logLik deviance Chisq Chi Df Pr(>Chisq)
#> EPP_ZIP_0$model  3 847.63 857.42 -420.82  841.63
#> EPP_ZIP$model    5 836.35 852.67 -413.18  826.35 15.279    2 0.0004811 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The result of the comparison confirms a significant difference between the yearly means, and the convenience of the concordance setting.

Next, let us apply the function `plot_BA` to generate the Bland-Altman plot and the bar plot of the differences in EPP between years. The plots are shown in Figures 4a and 4b.

```
EPP.BA<-plot_BA(EPP,y="Social",id="id",rm="Year") # Bland-Altman plot
```

```
plot_BA(EPP,y="Social",id="id",type="bars") # Bar plot
```

It can be seen that the magnitude of the differences grows as the mean increases. This heteroscedastic pattern is expected in counts because of the relation between the within-subjects variance and mean. Furthermore, we can compute some descriptive statistics of the differences:

```
summary(EPP.BA$data$Diff)
```

```
#>   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -7.0000 -1.0000  1.0000  0.9737  3.0000 10.0000
```

```
quantile(EPP.BA$data$Diff,probs=c(0.05,0.95))
```

```
#>  5% 95%
#> -4  6
```

Briefly, the mean of the differences between years is 0.97 fledglings, and the median is 1 fledgling. Ninety percent of the differences are between -4 and 6 fledglings between years.

CD34+ count cell example

The dataset *AF* includes data where a new method of flow cytometry for counting CD34+ cells is compared to the readings obtained by a standard approach (Fornas et al., 2000). Both methods (coded as 1 and 3 in the dataset) were applied to a sample of 20 subjects. The aim of the study is to assess

the interchangeability between the methods so it is necessary to evaluate the degree of concordance. Hence, the ICC used here concurs with the concordance correlation coefficient estimated by variance components (Carrasco and Jover 2003).

Let's firstly consider the Poisson model. As we are facing a concordance analysis, in the `icc_counts` function the name of the method's variable (`met`) has to be provided along with the counts variable (`y`) and subject's identifier (`id`). Additionally, it is necessary to specify the concordance analysis in the `type` argument because the default is the repeatability analysis.

```
AF_P <- icc_counts(AF, y = "y", id = "id", met = "met", type = "con")
ICC(AF_P)
```

```
#>          ICC    SE ICC 95% CI LL 95% CI UL
#> [1,] 0.8472696 0.021989 0.7982025 0.8851678
```

```
VarComp(AF_P)
```

```
#>      mu    BSVar    BMVar
#> 761.809 1.234619 0.1199439
```

The function `ICC` applied to the `iccc` object shows that the ICC estimate is 0.85 (95% confidence interval: 0.80, 0.89). Moreover, the function `VarComp` gives the parameters involved in the ICC estimator. In this case are the overall mean (`mu`), the between-subjects variance (`BSVar`) and the between-methods variability (`BMVar`) (the two last in log-scale).

Next, let's check the validity of the model by applying the function `GOF_check` to the `iccc` object.

```
set.seed(100)
AF_P.gof <- GOF_check(AF_P)
```

Figure 2a shows the plot of RQR with envelopes generated by simulation. Points on the plot stand for the RQR from the original sample. Most of points lie outside of the envelopes indicating the unsuitable fit of the model. Next plot (Figure 2b) shows the density of the RQR variances computed at the simulated samples. The RQR variance from the initial sample is 32.2 which is much larger than those from the simulations. The p-value to test overdispersion is obtained by applying the function `DispersionTest` to the `GOF`. With regards the zero inflation, no zeros were found in data so it is unnecessary to check this issue.

```
DispersionTest(AF_P.gof)
```

```
#>      S    P_value
#> 32.20049 0.00990099
```

Consequently, it is necessary to use a model able to afford the overdispersion as the Negative Binomial distribution.

```
AF_NB2 <- icc_counts(AF, y = "y", id = "id", met = "met", type = "con", fam = "nbinom2")
ICC(AF_NB2)
```

```
#>          ICC    SE ICC 95% CI LL 95% CI UL
#> [1,] 0.834794 0.0454062 0.7212048 0.9046669
```

```
VarComp(AF_NB2)
```

```
#>      mu    BSVar    BMVar      r
#> 777.1946 1.188904 0.0809433 0.0488122
```

In this case, the ICC is quite similar to that from the Poisson model (0.83, 95% CI: 0.72, 0.90) but the confidence interval is wider. The ICC components are the same as those from the Poisson case (with different values) plus the Negative Binomial dispersion parameter (0.049). Concerning the fit of the model,

```
set.seed(100)
AF_NB2.gof <- GOF_check(AF_NB2)
```

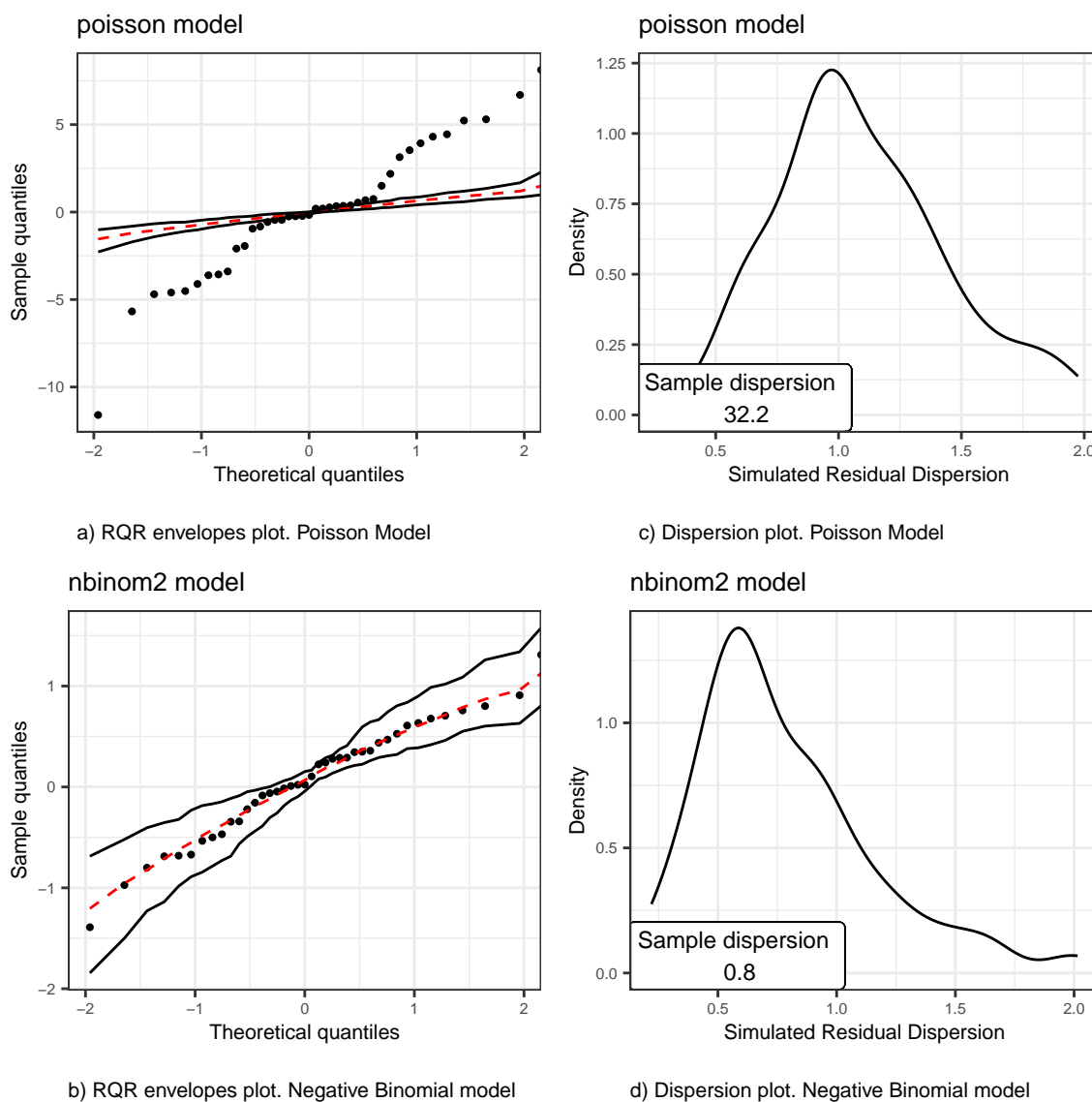


Figure 2: Goodness of fit for CD34 cell count example. The Randomized Quantile Residuals (RQR) of original data are compared to those from simulated data under the fitted model. The plots shown are RQR with envelopes, and dispersion of RQR. First row shows results for Poisson model while the plots for Negative Binomial model are on second row.


```
plot(AF_NB2.gof)
```

in Figure 2c all the RQR are within the envelopes indicating the appropriateness of the model. Additionally, the dispersion from the initial sample is within the expected values (Figure 2d). This result is also confirmed by running the dispersion test:

```
DispersionTest(AF_NB2.gof)
```

```
#>          S    P_value
#> 0.8002765 0.4059406
```

Figures 4c and 4d show the Bland-Altman plot and the Bar plot of the differences between method 1 and 2 that are generated by the following commands:

```
AF.BA <- plot_BA(AF,y="y",id="id", rm="met") # Bland-Altman plot
```

```
plot_BA(AF,y="y",id="id", type="bars") # Bar plot
```

It can be seen that for values of the mean below 700 the within-subjects differences are very close to 0. However, for larger values of the mean there is a trend in the differences in relation to the mean values.

Tick counts example

In this study, the repeatability of line transects survey method to estimate tick abundance was assessed (Kjellander et al. 2021) in the area of Grimso (Sweden). With this aim, sampling was performed by two parallel transects separated by 1m-2m where the total count of ticks was recorded. Here, the transects are the cluster variable and every pair of data from the same transect are considered as replicates. Data is stored in the package as the *Grimso* object.

As seen before the first model to consider is the Poisson model. As it is a repeatability analysis, in the *icc_counts* function we just need to provide the name of the counts variable (*Tot*) and subject's identifier (*TransectID*).

```
G_P <- icc_counts(Grimso, y = "Tot", id = "TransectID")
ICC(G_P)
```

```
#>          ICC    SE ICC 95% CI LL 95% CI UL
#> [1,] 0.3494333 0.1369518 0.0589753 0.5853431
```

```
VarComp(G_P)
```

```
#>          mu    BSVar
#> 0.2072297 1.278685
```

The function *ICC* applied to the *iccc* object shows that the ICC estimate is 0.35 (95% confidence interval: 0.06, 0.59). The function *VarComp* gives the parameters involved in the ICC estimator: the overall mean and the between-subjects variance (the latter in log-scale).

Let's apply the function *GOF_check* to the *iccc* object to check the validity of the model.

```
set.seed(100)
G_P.gof <- GOF_check(G_P)
```

```
plot(G_P.gof)
```

Figure 3a shows the plot of RQR with envelopes generated by simulation. All points on the plot lie within the envelopes indicating the fit of the model is correct. Additionally, Figure 3b shows the RQR variance from the initial sample (1.84) is compatible with the dispersion observed in the simulated samples. The overdispersion test is run by applying the function *DispersionTest* to the *GOF* object.

```
DispersionTest(G_P.gof)
```

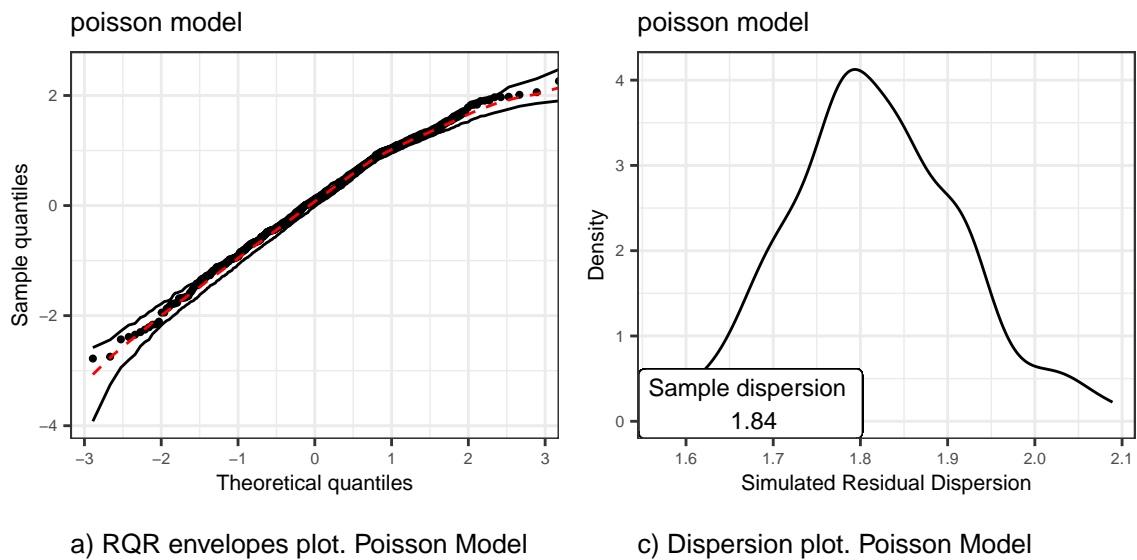


Figure 3: Goodness of fit for Tick counts example. The Randomized Quantile Residuals (RQR) of original data are compared to those from simulated data under the fitted model. The plots shown are RQR with envelopes, and dispersion of RQR for Poisson model.

```
#>      S   P_value
#> 1.83724 0.4158416
```

The p-value is 0.416 so the null hypothesis of no overdispersion is not rejected and no further models have to be fitted.

The Bland-Altman plot and the Bar plot of the within-subjects differences are shown in Figures 4e and 4f.

```
G.BA <- plot_BA(Grimso,y="Tot",id="TransectID",rm="Round") # Bland-Altman plot
```

```
plot_BA(Grimso,y="Tot",id="TransectID", type="bars") # Bar plot
```

As in the case of the sparrow fledgling paternity counts, we can observe a heteroscedastic pattern with the variability of the differences increasing with the mean. Most of the differences are 0 (75%) and 90% of the differences lie between -1 and 1.

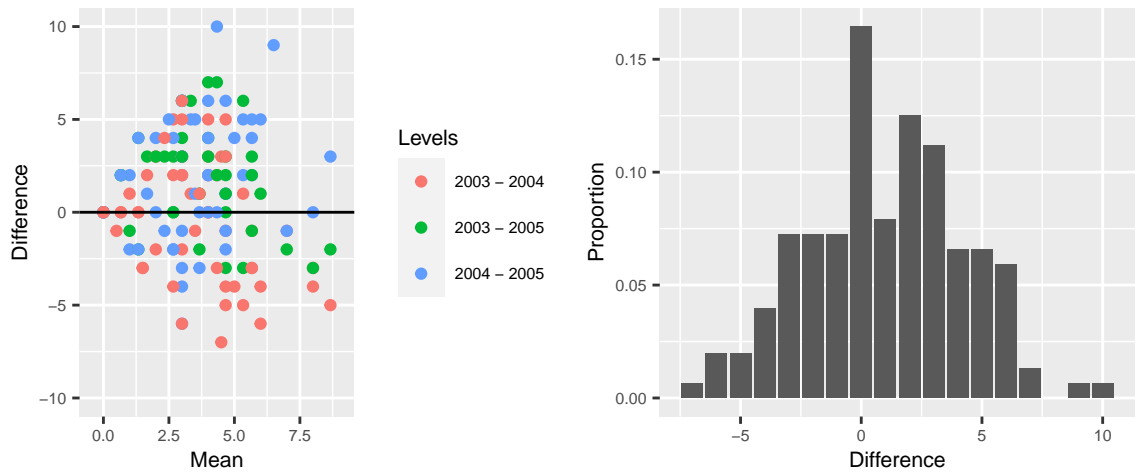
```
quantile(G.BA$data$Diff, probs=c(0.05,0.95))
```

```
#> 5% 95%
#> -1  1
```

5 Conclusion

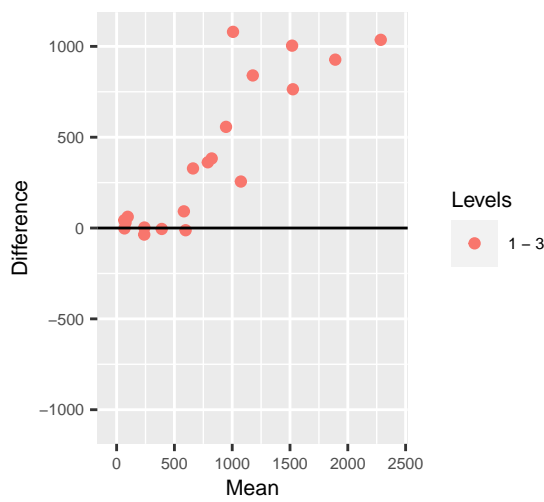
The statistical assessment of agreement is an issue that has received a considerable attention in recent years. It is possible to find statistical software to carry out such analysis for qualitative or continuous data (see for example Revelle (2021);Carrasco et al. (2013);D. Feng (2020)). However, there is a lack of tools when dealing with discrete data. Here, the **iccCounts** package have been introduced to assess the agreement with such type of data considering both repeatability and concordance settings. Furthermore, the **iccCounts** package provides the methodology to assess the validity of the model fitted to data.

It is important to note that no factors or predictors other than subjects and methods have been considered in the linear predictor of the GLMM. When fitting a GLMM, the inclusion of further covariates allows controlling for confounding effects. In this case, the ICC computed after controlling for confounding effects is referred to as adjusted repeatability (Nakagawa and Schielzeth 2010). Including covariates in the linear predictor make sense when the aim is to estimate the magnitude of an effect (difference in means, odds ratio or ratio of means, for instance) adjusted by the covariates.

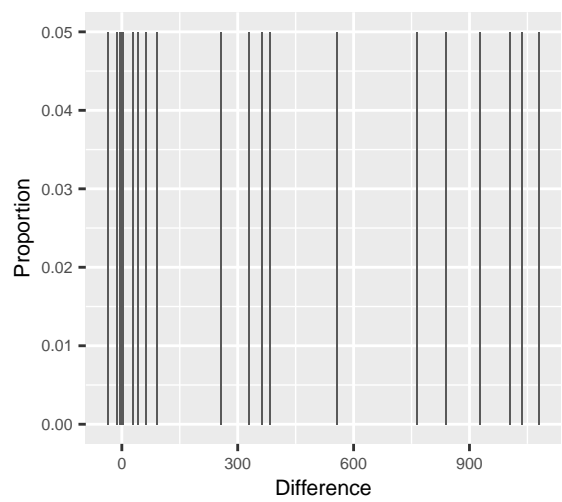


a) Bland-Altman plot. Sparrow fledglings paternity example

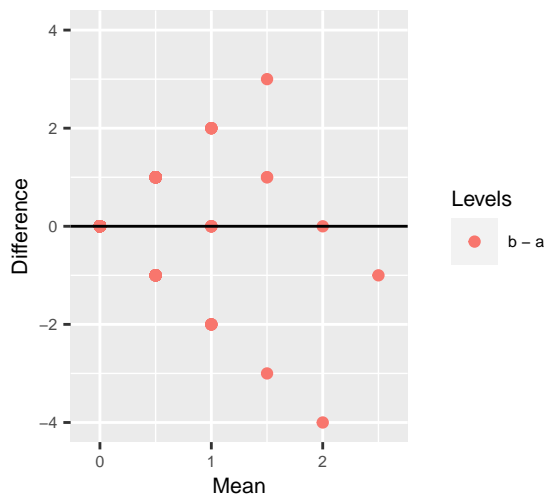
b) Bar plot of differences. Sparrow fledglings paternity example



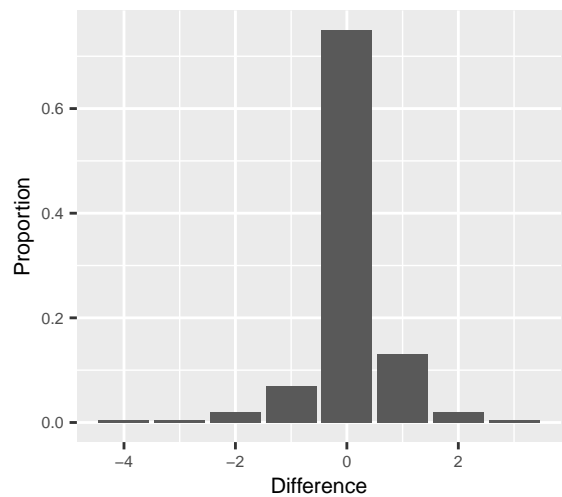
c) Bland-Altman plot. CD34+ count cell example



d) Bar plot of differences. CD34+ count cell example



e) Bland-Altman plot. Tick counts example



f) Bar plot of differences. Tick counts example

Figure 4: Bland-Altman and Bar plots. The first column shows the Bland-Altman plots where difference between pairs of data from the same subject (Y-axis) is plotted against mean of data from the same subject (X-axis). The second column contains the Bar plots of the differences between pairs of data from the same subject. The plots for Sparrow fledglings paternity example are on the first row, the CD34+ count cell example plots are on second row, and plots for Tick counts example are on third row.

When estimating the ICC in a linear model, the inclusion of covariates will lead to a change in the variance components estimates but they remain as common estimates for all subjects. However, when facing the models for count data, the addition of covariates in the linear predictor leads to different ICCs because the value of the marginal mean μ will be different for every level of the covariates. For this reason, in the case of count data, it is preferable to segregate the data to estimate a different ICC according to the covariates. In this way, both the variance components and the mean will be different.

Furthermore, in the Normal model setting the ICC takes values from 0 to 1. A value of 0 means independence among the measures from the same cluster (no cluster effect) whilst a value of 1 implies perfect agreement, i.e. all data from the same subject are equal. However, it is not possible to reach a value of 1 in the counts setting. The reason for this is that some within-subject variability is unavoidable because of the relation between the variance and the mean in these models. Thus, it is not possible to observe perfect agreement with count data but the interpretation remains the same: the proportion of the total variance accounted for between-subjects variability.

6 Availability

The current stable version of the package requires R 4.0 and can be downloaded from CRAN . Furthermore, *iccCounts* depends on the following R packages: *glmmTMB* (Brooks et al. 2017); *ggplot2* (Wickham 2016); *Deriv* (Clausen and Sokol 2020); *gridExtra* (Auguie 2017); and *dplyr* (Wickham et al. 2020).

7 Acknowledgments

I would like to thank to Shinichi Nakagawa from UNSW Sydney (Australia) for kindly providing the Sparrow fledglings paternity data. I am also indebted to Pia Kjellander from Linköping University (Sweden) for sharing the Tick count data.

References

- Auguie, B. 2017. *gridExtra: Miscellaneous Functions for "Grid" Graphics*. R Package Version 2.3. <https://CRAN.R-project.org/package=gridExtra>.
- Bland, J Martin, and Douglas G Altman. 1995. "Comparing Methods of Measurement: Why Plotting Difference Against Standard Method Is Misleading." *The Lancet* 346 (8982): 1085–87.
- Bolker, Benjamin M., Mollie E. Brooks, Connie J. Clark, Shane W. Geange, John R. Poulsen, M. Henry H. Stevens, and Jada-Simone S. White. 2009. "Generalized Linear Mixed Models: A Practical Guide for Ecology and Evolution." *Trends in Ecology & Evolution* 24 (3): 127–35.
- Brooks, ME, K Kristensen, KJ van Benthem, A Magnusson, CW Berg, A Nielsen, HJ Skaug, M Maechler, and BM Bolker. 2017. "glmmTMB Balances Speed and Flexibility Among Packages for Zero-Inflated Generalized Linear Mixed Modeling." *The R Journal* 9: 299–314.
- Carrasco, Josep L. 2010. "A Generalized Concordance Correlation Coefficient Based on the Variance Components Generalized Linear Mixed Models for Overdispersed Count Data." *Biometrics* 66: 897–904.
- Carrasco, Josep L., and L Jover. 2003. "Estimating the Generalized Concordance Correlation Coefficient Through Variance Components." *Biometrics* 59: 849–58.
- . 2005. "Concordance Correlation Coefficient Applied to Discrete Data." *Statistics in Medicine* 24: 4021–34.
- Carrasco, Josep L., BR Phillips, J Puig-Martinez, TS King, and VM Chinchilli. 2013. "Estimation of the Concordance Correlation Coefficient for Repeated Measures Using SAS and R." *Computer Methods and Programs in Biomedicine* 109: 293–304.
- Choudhary, PK, and HN Nagaraja. 2017. *Measuring Agreement: Models, Methods, and Applications*. Hoboken: Wiley.
- Clausen, A, and S Sokol. 2020. *Deriv: R-Based Symbolic Differentiation*. *Deriv Package Version 4.1*. <https://CRAN.R-project.org/package=Deriv>.
- DeVet, HCW, CB Terwee, LB Mokkink, and DL Knol. 2011. *Measurement in Medicine*. Cambridge: Cambridge University Press.
- Dunn, PK, and GK Smyth. 1996. "Randomized Quantile Residuals." *Journal Computational and Graphical Statistics* 5: 236–44.
- Feng, C, L Li, and A Sadeghpour. 2020. "A Comparison of Residual Diagnosis Tools for Diagnosing Regression Models for Count Data." *BMC Medical Research Methodology* 20: 175.

- Feng, Dai. 2020. *agRee: Various Methods for Measuring Agreement*. <https://CRAN.R-project.org/package=agRee>.
- Fleiss, JL. 1986. *The Design and Analysis of Clinical Experiments*. New York: Wiley.
- Hardin, W, and JM Hilbe. 2007. *Generalized Linear Models and Extensions*. Stata Press.
- Hoef, JM Ver. 2012. "Who Invented the Delta Method?" *The American Statistician* 66: 124–27.
- Kjellander, PL, M Aronsson, UA Bergvall, Josep L. Carrasco, M Christensson, PE Lindgren, M Akesson, and P Kjellander. 2021. "Validating a Common Tick Survey Method: Cloth-Dragging and Line Transects." *Experimental and Applied Acarology* 83: 131–46.
- Nakagawa, S, and H Schielzeth. 2010. "Repeatability for Gaussian and Non-Gaussian Data: A Practical Guide for Biologists." *Biological Reviews* 85: 935–56.
- Revelle, William. 2021. *Psych: Procedures for Psychological, Psychometric, and Personality Research*. Evanston, Illinois: Northwestern University. <https://CRAN.R-project.org/package=psych>.
- Schroeder, J, T Burke, ME Mannarelli, DA Dawson, and S Nakagawa. 2012. "Maternal Effects and Heritability of Annual Productivity." *Journal of Evolutionary Biology* 25: 149–56.
- Smith, Mark W., Jun Ma, and Randall S. Stafford. 2010. "Bar Charts Enhance Bland–Altman Plots When Value Ranges Are Limited." *Journal of Clinical Epidemiology* 63 (2): 180–84.
- Stoffel, MA, S Nakagawa, and H Schielzeth. 2017. "rptR: Repeatability Estimation and Variance Decomposition by Generalized Linear Mixed-Effects Models." *Methods in Ecology and Evolution* 8: 1639–44.
- Wickham, H. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. New York: Springer-Verlag.
- Wickham, H, R François, L Henry, and K Müller. 2020. *Dplyr: A Grammar of Data Manipulation*. R Package Version 1.0.2. <https://CRAN.R-project.org/package=dplyr>.
- Wolak, ME, JF Fairbairn, and RP Paulsen. 2012. "Guidelines for Estimating Repeatability." *Methods in Ecology and Evolution* 3: 129–37.

Josep L. Carrasco

University of Barcelona

Biostatistics. Department of Basic Clinical Practice

Casanova 143, 08036, Barcelona, Spain

https://webgrec.ub.edu/webpages/personal/ang/005037_jlcarrasco.ub.edu.html

ORCID: 0000-0003-1184-0753

jlcarrasco@ub.edu

R-miss-tastic: a unified platform for missing values methods and workflows

by Imke Mayer, Aude Sportisse, Julie Josse, Nicholas Tierney and Nathalie Vialaneix

Abstract Missing values are unavoidable when working with data. Their occurrence is exacerbated as more data from different sources become available. However, most statistical models and visualization methods require complete data, and improper handling of missing data results in information loss or biased analyses. Since the seminal work of Rubin (1976), a burgeoning literature on missing values has arisen, with heterogeneous aims and motivations. This led to the development of various methods, formalizations, and tools. For practitioners, however, it remains a challenge to decide which method is most appropriate for their problem, in part because this topic is not systematically covered in statistics or data science curricula. To help address this challenge, we have launched the ‘R-miss-tastic’ platform, which aims to provide an overview of standard missing values problems, methods, and relevant implementations of methodologies. Beyond gathering and organizing a large majority of the material on missing data (bibliography, courses, tutorials, implementations), ‘R-miss-tastic’ covers the development of standardized analysis workflows. Indeed, we have developed several pipelines in R and Python to allow for hands-on illustration of and recommendations on missing values handling in various statistical tasks such as matrix completion, estimation, and prediction, while ensuring reproducibility of the analyses. Finally, the platform is dedicated to users who analyze incomplete data, researchers who want to compare their methods and search for an up-to-date bibliography, and teachers who are looking for didactic materials (notebooks, recordings, lecture notes).

1 Context and motivation

Missing data are unavoidable as soon as collecting or acquiring data is involved. They occur for many reasons including: individuals choosing not to answer survey questions, measurement devices failing, or data having simply not been recorded. Their presence becomes even more important as data are now obtained at increasing velocity and volume, and from heterogeneous sources not originally designed to be analyzed together. As pointed out by Zhu et al. (2019), “one of the ironies of working with Big Data is that missing data play an ever more significant role, and often present serious difficulties for analysis”. Despite this, the approach most commonly implemented by default in software is to toss out cases with missing values. At best, this is inefficient because it wastes information from the partially observed cases. At worst, it results in biased estimates, particularly when the distributions of the missing values are systematically different from those of the observed values (e.g., Enders, 2010, Chap. 2).

However, handling missing data in a more efficient and relevant way (than limiting the analysis to solely the complete cases) has attracted a lot of attention in the literature in the last two decades. In particular, a number of reference books have been published (Schafer and Graham, 2002; van Buuren, 2018; Carpenter and Kenward, 2012; Little and Rubin, 2019) and the topic is an active field of research (Josse and Reiter, 2018). The diversity of the missing data problems means there is great variety in the proposed and studied methods. They include model-based approaches, integrating likelihoods or posterior distributions over missing values, filling in missing values in a realistic way with single, or multiple imputations, or weighting of observations, appealing to ideas from the design-based literature in survey sampling. The multiplicity of the available solutions makes sense because there is no single solution or tool to manage missing data: the appropriate methodology to handle them depends on many features, such as the objective of the analysis, the type of data, the type of missing data and their pattern. Some of these methods are available in various software solutions. As R is one of the main pieces of software for statisticians and data scientists, with its development starting almost three decades ago (Ihaka, 1998), R offers the largest number of implemented approaches. This is also due to its ease in incorporating new methods and its modular packaging system. Currently, there are over 270 R packages on CRAN that mention missing data or imputation in their DESCRIPTION files. These packages serve many different applications, data types or types of analysis. More precisely, exploratory and visualization tools for missing data are available in packages like **naniar**, **VIM**, and **MissingDataGUI** (Tierney et al., 2021; Tierney and Cook, 2018; Kowarik and Templ, 2016; Cheng et al., 2015). Imputation methods are included in packages like **mice**, **Amelia**, and **mi** (van Buuren and Groothuis-Oudshoorn, 2011; Honaker et al., 2011; Gelman and Hill, 2011). Other packages focus on dealing with complex, heterogeneous (categorical, quantitative, ordinal variables) data or with large dimensional multi-level data, such as **missMDA**, and **MixedDataImpute** (Josse et al., 2016; Murray and Reiter, 2015). Besides R, other languages such as Python (Van Rossum and Drake, 2009), which currently only have few publicly available implementations of methods that handle missing values,

offer increasingly more solutions. Two prominent examples are: 1) the **scikit-learn** library (Pedregosa et al., 2011) which has recently added a module for missing values imputation; and 2) the **DataWig** library (Biessmann et al., 2018) which provides a framework to learn to impute incomplete data tables.

Despite the large range of options, missing data are often not handled appropriately by practitioners. This may be for several reasons. First, the plethora of options can be a double-edged sword; while it is great to have many options, finding the most appropriate method is challenging as there are so many. Second, the topic of missing data is often missing itself from many statistics and data science syllabuses, despite its omnipresence in data. So, when faced with missing data, practitioners are left powerless; quite possibly never having been taught about missing data, they do not know how to approach the problem, the dangers of mismanagement, how to navigate the methods, software, or how to choose the most appropriate method or workflow.

To help promote better management and understanding of missing data, we have released ‘R-miss-tastic’, an open platform for missing values. The platform takes the form of a reference website¹, which collects, organizes and produces material on missing data. It has been conceived by an infrastructure steering committee working group (ISC; its members are authors of this article), which first provided a CRAN Task View² on missing data³ that lists and organizes existing R packages on the topic. The ‘R-miss-tastic’ platform extends and builds on the CRAN Task View by collecting, creating and organizing articles, tutorials, documentation, and workflows for analyses with missing data.

This platform is easily extendable and well documented, allowing it to seamlessly incorporate future works and research in missing values. The intent of the platform is to foster a welcoming community, within and beyond the R community. ‘R-miss-tastic’ has been designed to be accessible for a wide audience with different levels of prior knowledge, needs, and questions. This includes students, teachers, statisticians, and researchers. Students can use its content as complementary course material. Teachers can use it as a reference website for their own classes. Statisticians and researchers can find example analysis workflows, or even contribute information for specific areas and find collaborators.

The platform provides new tutorials, examples and pipelines of analyses that we have developed with missing data spanning the entirety of an analysis. These have been developed in R and in Python, implementing standard methods for generating missing values, and for analyzing them under different perspectives. In addition, we reference publicly available datasets that are commonly used as benchmarks for new missing values methodologies. The developed pipelines cover the entirety of a data analysis: exploratory analyses, establishing statistical and machine learning models, analysis diagnostics, and finally interpreting results obtained from incomplete data. We hope these pipelines also serve as a guide when choosing a method to handle missing values.

The remainder of the article is organized as follows: In the section entitled “Structure and content of the platform” we describe the different components of the platform, the structure that has been chosen, and the target audience. The section is organized as the platform itself, starting by describing materials for less advanced users then materials for researchers and finally resources for practical implementation. We then detail the implementation and use-cases of the provided R and Python workflows in the following section entitled “Details of the missing values workflows”. Finally, in the conclusion, we outline an overview of planned future developments for the platform and interesting areas in missing values research that we would like to bring to a wider audience.

2 Structure and content of the platform

The ‘R-miss-tastic’ platform is released at <https://rmissstastic.netlify.com/>. It has been developed using the R package **blogdown** (Xie et al., 2017) which generates static websites using Hugo⁴. Live examples have been included using the tool <https://rdr.io/snippets/> provided by the website ‘R Package Documentation’. The source code and materials of the platform have been made publicly available on GitHub at <https://github.com/R-miss-tastic>, which provides a transparent record of the platform’s development, and facilitates contributions from the community.

We now discuss the structure of the ‘R-miss-tastic’ platform, the aim and content of each subsection, and highlight key features of the platform.

¹<https://rmissstastic.netlify.com/>

²<https://CRAN.R-project.org/package=ctv>

³<https://cran.r-project.org/web/views/MissingData.html>

⁴<https://gohugo.io/>

Missing values workflows

An important contribution and novelty of this work is the proposal of several workflows that allow for a hands-on illustration of classical analyses with missing values, both on simulated data and on publicly available real-world data. These workflows are provided both in R and in Python code and cover the following topics:

- *How to generate missing values?* Generate missing values under different mechanisms, on complete or incomplete datasets. This is useful when performing simulations to compare methods that impute or handle missing data.
- *How to do statistical inference with missing values?* In particular, we focus on different solutions for estimating linear and logistic regression parameters with missing covariate values (maximum likelihood or multiple imputation).
- *How to impute missing values?* We compare different single imputation/matrix completion methods (e.g., using conditional models, low-rank models, etc.).
- *How to predict with missing values?* We consider building predictive models, e.g., using random forests (Breiman, 2001), on data with incomplete predictors. The workflows present different strategies to deal with missing values in the covariates both in the training set and in the test set.

The aim of these workflows is threefold: 1) they provide a practical implementation of concepts and methods discussed in the lectures and bibliography sections of the platform; 2) they are implemented in a generic way, allowing for re-use on other datasets, for integration of other estimation or imputation methods; 3) the distinction between inference, imputation, and prediction lets the user keep in mind the solutions are not the same.

Furthermore, the workflows allow for a transparent and open discussion about the proposed implementations, which can be followed on the project GitHub repository, referencing proposals and discussions about practicable extensions of the workflows.

Additionally, a workflow on *How to do causal inference with incomplete covariates/attributes in R?* demonstrates simple weighting and doubly robust estimators for treatment effect estimation using R. This workflow is based on the R implementation of the methodology proposed by Mayer et al. (2020).

We provide a more detailed view on the proposed workflows in a later section, with examples of tabular or graphical outputs that can be obtained as well as recommendations on how to interpret and leverage these outputs.

Missing values lectures

For someone unfamiliar with missing data, it is a challenge to know where to begin the journey of understanding them, and the methods to handle them. This challenge is addressed with ‘R-miss-tastic’, which makes the material to get started easily accessible.

Teaching and workshop material takes many forms – slides, course notes, lab workshops, video tutorials and in-depth seminars. The material is of high quality, and has been generously contributed by numerous renowned researchers who investigate the problems of missing values, many of whom are professors having designed introductory and advanced classes for statistical analyses with missing data. This makes the material on the ‘R-miss-tastic’ platform well suited for both beginners and more experienced users.

These teaching and workshop materials are described as “lectures”, and are organized into five sections:

1. General lectures: Introduction to statistical analyses with missing values; the role of visualization and exploratory data analysis for understanding missingness and guiding its handling; theory and concepts are covered, such as missing values mechanisms, likelihood methods, and imputation.
2. Multiple imputation: Introduction to popular methods of multiple imputation (joint modeling and fully conditional), how to correctly perform multiple imputation and limits of imputation methods.
3. Principal component methods: Introduction to methods exploiting low-rank type structures in the data for visualization, imputation and estimation.
4. Specific data or applications types: Lectures covering in details various sub-problems such as missing values in *time series*, in *surveys*, or in treatment effect estimation (*causal inference*). Indeed, certain data types require adaptations of standard missing values methods (e.g., handling time dependence in time series (Moritz and Bartz-Beielstein, 2017)) or additional assumptions about

R-miss-tastic

A resource website on missing values - Methods and references for managing missing data

Below you will find a selection of high-quality lectures, tutorials and labs on different aspects of missing values.

If you wish to contribute some of your own material to this platform, please feel free to contact us via the [Contact form](#).

[General lectures](#)

[Multiple imputation](#)

[Missing values and principal component methods](#)

[Specific data or application types](#)

[Implementation in R](#)

Figure 1(a) shows a collapsed view of the website with a 'Collapse All' button and expand/collapse icons for each topic. Figure 1(b) shows an expanded view of the 'General tutorials' section, displaying a list of lecture titles and their authors, such as 'Statistical Methods for Analysis With Missing Data' by Mauricio Sadinle and 'Analysis of missing values' by Jae-Kwang Kim.

(a) Collapsed view

(b) Extract

Figure 1: Overview of selected lectures available on the platform (shared by various researchers working on missing values). The lectures are grouped by topic. Each lecture contains slides or lecture notes and in certain cases also comes with corresponding R tutorials and lecture recordings.

the impact of missing values (such as the impact on confounded treatment effects in causal inference (Mayer et al., 2020)). More in-depth material, e.g., video recordings from a virtual workshop on *Missing Data Challenges in Computation, Statistics and Applications*⁵ held in 2020, is also available.

5. Implementations: A non-exhaustive list of detailed vignettes describing functionalities of R packages and of Python modules that implement some of the statistical analysis methods covered in the other lectures. For example, the functionalities and possible applications of the **missMDA** R package are presented in a brief summary, allowing the reader to compare the main differences between this package and the **mice** package which is also summarized using the same summary format.

Figure 1 illustrates two views of the lectures page: Figure 1(a) shows a collapsed view presenting the different topics, Figure 1(b) shows an example of the expanded view of one topic (General tutorials), with a detailed description of one of the lectures (obtained by clicking on its title), 'Analysis of missing values' by Jae-Kwang Kim. Each lecture can contain several documents (as is the case for this one) and is briefly described by a header presenting its purpose.

Lectures that we found very complete and thus highly recommend are:

- *Statistical Methods for Analysis with Missing Data* by Mauricio Sadinle (in 'General tutorials');
- *Missing Values in Clinical Research – Multiple Imputation* by Nicole Erler (in 'Multiple imputation');
- *Handling missing values in PCA and MCA* by François Husson. (in 'Missing values and principal component methods');
- *Modern use of Shared Parameter Models for Dropout (in longitudinal and time-to-event data)* by Dimitris Rizopoulos (in 'Specific data or application types').

The purpose of these lectures is to provide either an introduction or a deeper understanding of the statistical problems and proposed solutions in terms of their (mathematical) derivation and theoretical scope. So there is less of a focus on practical demonstrations with real data, or a systematic comparison of all methods for the same problems. This is covered in the section presenting in detail the developed workflows.

References on missing values

Complementary to the *Lectures* section, this part of the platform serves as a broad overview on the scientific literature discussing missing values taxonomies and mechanisms and statistical, machine learning methods to handle them. This overview covers both classical references to books, articles, etc. such as Schafer and Graham (2002); van Buuren (2018); Carpenter and Kenward (2012); Little and Rubin (2019) and more recent developments such as Josse et al. (2019); Gondara and Wang (2018),

⁵<https://www.ias.edu/math/mdccsa>

R-miss-tastic

A resource website on missing values - Methods and references for managing missing data

On this platform we attempt to give you an overview of main references on missing values. We do not claim to gather all available references on the subject but rather to offer a peak into different fields of active research on handling missing values, allowing for an introductory reading as well as a starting point for further bibliographical research.

[See here for a full \(and uncommented\) list of references.](#)

Inspired by [CRAN Task View on Missing Data](#) and a [review](#) of Imbert & Villa-Vieñe on handling missing values (2016, written in French) we organized our selection of relevant references on missing values by different topics.

[Short introduction to missing values](#)

[General references and reviews](#)

[Weighting methods](#)

[Hot-deck and kNN approaches](#)

[Likelihood-based approaches](#)

[Single imputation](#)

[Multiple imputation](#)

[Machine Learning](#)

[Missing values mechanisms](#)

(a) Curated version

R-miss-tastic

A resource website on missing values - Methods and references for managing missing data

[A commented version of this bibliography can be found here.](#)

Publication type	Year	Author	
All	All	Search by author name...	
<hr/>			
Citation	Year	Publication type	
Collapse All			
Abayomi, K., A. Gelman, and M. Levy. <i>Diagnostics for multivariate imputations</i> . In: <i>Journal of the Royal Statistical Society, Series C (Applied Statistics)</i> 57.3 (2008), pp. 273-281.	2008	Article	DOI
Albert, P. S. and D. A. Follmann. <i>Modeling repeated count data subject to informative dropout</i> . In: <i>Biometrics</i> 56.3 (2000), pp. 667-677.	2000	Article	DOI
Allison, P. D. <i>Missing Data, Quantitative Applications in the Social Sciences</i> . Thousand Oaks, CA, USA: Sage Publications, 2001. ISBN: 9780761916727.	2001	Book	DOI
Andridge, R. and R. J. A. Little. <i>A review of hot-deck imputation for survey non-response</i> . In: <i>International Statistical Review</i> 78.1 (2010), pp. 40-64.	2010	Article	DOI
Audigier, V., F. Husson, and J. Josse. <i>A principal component method to impute missing values for mixed data</i> . In: <i>Advances in Data Analysis and Classification</i> 10.1 (2016), pp. 6-26.	2016	Article	DOI
Audigier, V., F. Husson, and J. Josse. <i>MIMCA: multiple imputation for categorical variables with multiple correspondence analysis</i> . In: <i>Statistics and Computing</i> 27.2 (2016), pp. 1-18. eprint: 1505.08116.	2016	Article	DOI
Collapse All			

(b) Alphabetical version

Figure 2: Overview of curated (a) and alphabetical (b) bibliographies on missing values problems and methods.

which study the consistency of supervised learning with missing values. The entire (non-exhaustive) bibliography can be browsed in two ways: 1) a complete list, filtered by publication type and year, with a search option for the authors or, 2) a curated version. For 2), we classified the references into several domains of research or application, briefly discussing important aspects of each domain. This dual representation is shown in Figure 2 and allows for an extensive search in the existing literature, while providing some guidance for those focused on a specific topic. All references are also collected in a unique BibTex file made available in the GitHub repository⁶. This shared file allows external users to easily propose additions to the bibliography, which are then reviewed by the platform committee, composed of researchers with different focuses on missing values.

Missing values implementations

R packages As mentioned in the introduction, the platform development is based on the release of the *MissingData* CRAN Task View, which currently lists approximately 150 R packages. The CRAN Task View is continuously updated, adding new, and removing obsolete R packages. Packages are organized by topic: *exploration of missing data*, *likelihood based approaches*, *single imputation*, *multiple imputation*, *weighting methods*, *specific types of data*, and *specific application fields*. We selected only sufficiently mature and stable packages already published on CRAN or Bioconductor. This ensures all listed packages can easily be installed and used by practitioners.

Even though the Task View classifies packages into different sub-domains, it may still be a challenge for practitioners and researchers inexperienced with missing values to choose the most relevant package for their application. To address this challenge, we provide a partial and slightly more detailed overview of existing R packages, selecting the most popular and versatile ones. This overview is a blend of the Task View, and of the individual package description pages and vignettes as provided on CRAN or Bioconductor. For each selected package (7 at the date of writing of this article: **imputeTS**, **mice**, **missForest**, **missMDA**, **naniar**, **simputation** and **VIM**), we provide a category (in the style of the categorization in the Task View), a short description of use-cases, its description (as on CRAN), the usual CRAN statistics (number of monthly downloads, last update), the handled data formats (e.g., `data.frame`, `matrix`, `vector`), a list of implemented algorithms (e.g., k-means, PCA, decision tree), the list of available datasets, some references (such as articles and books), and a small chunk of code, ready for a direct execution on the platform via the *R package Documentation*⁷. Figure 3 shows the condensed view of the package page and the expanded description sheet of a given package (here **naniar**).

We believe shortlisting R packages is highly useful for practitioners new to the field, as it demonstrates data analysis that handles missing values in the data. We are aware that this selection is subjective, and we welcome external suggestions for other packages to add to this shortlist.

⁶in resources/rmisstastic_biblio.bib

⁷<https://rdrr.io/snippets/>

R-miss-tastic

A resource website on missing values - Methods and references for managing missing data

R Packages

This page provides introductions to popular missing data packages with small examples on how to use them. Thus the page gives more extensive information than the [CRAN Task View on Missing Data](#), which is recommended to get a first overall overview about the CRAN missing data landscape.

You can also contribute on your own to this page and provide a short introduction to a missing data package. Take a look at [this short description](#) on how to do this. We are very happy about all contributions.

Search Sort by name Sort by Category

- **missMDA**

Category: Single and Multiple Imputation, Multivariate Data Analysis
Imputation of incomplete continuous or categorical datasets; Missing values are imputed with a principal component analysis (PCA), a multiple correspondence analysis (MCA) model or a multiple factor analysis (MFA) model; Perform multiple imputation with and in PCA or MCA.

downloads: 4000/month CRAN 2019-01-23
[DOI](#)

- **imputeTS**

Category: Time-Series Imputation, Visualisations for Missing Data
Imputation (replacement) of missing values in univariate time series. Offers several imputation functions and missing data plots. Available imputation algorithms include: 'Mean', 'LOCF', 'Interpolation', 'Moving Average', 'Seasonal Decomposition', 'Kalman Smoothing on Structural Time Series models', 'Kalman Smoothing on ARIMA models'.

downloads: 12K/month CRAN 2019-07-01
[DOI](#)

- **mice**

Category: Multiple Imputation
Multiple imputation using Fully Conditional Specification (FCS) implemented by the MICE algorithm as described in Van Buuren and Groothuis-Oudshoorn (2011). Each variable has its own imputation model. Built-in imputation models are provided for continuous data (predictive mean matching, normal), binary data (logistic regression), unordered categorical data (polytomous logistic regression) and ordered categorical data (proportional odds). MICE can also impute continuous two-level data (normal model, pan, second-level variables). Passive imputation can be used to maintain consistency between variables. Various diagnostic plots are available to inspect the quality of the imputations.

downloads: 41K/month CRAN 2019-07-10
[DOI](#)

(a) Extract of global view

Package:

naniar

Category:

Data Structures, Summaries, and Visualisations for Missing Data

Use-Cases:

Visualization of missing values, descriptive statistics, ...

Popularity:

downloads: 230/month

Description:

Missing values are ubiquitous in data and need to be carefully explored and handled in the initial stages of analysis. In this vignette we describe the tools in the package naniar for exploring missing data structures with minimal deviation from the common workflows of ggplot and tidy data.

Last update:

CRAN 2019-03-15

Datasets:

- oceanbuoys
- pedestrian
- riskfactors

Further Information:

- Tierney, N. J., & Cook, D. H. (2018). Expanding tidy data principles to facilitate missing data exploration, visualization and assessment of imputations. *arXiv preprint arXiv:1809.02264*. [PDF](#) [on arXiv](#)
- [Vignettes](#)
- Related [jvsdat](#) R-package

Input:

data.frame, vector

Example:

```
library(naniar)
data(airquality)
print("print data set with NAs")
print(head(airquality))

## Replace "NA" values with values 10% lower
## than the minimum value in that variable.
## This is done by calling the geom_miss_point() function
ggplot2::ggplot(airquality,
                 ggplot2::aes(x = Solar.R,
                              y = Ozone)) +
  geom_miss_point()
```

Here you can have an interactive look at the example:

```
library(naniar)
data(airquality)
print("print data set with NAs")
print(head(airquality))

## Replace "NA" values with values 10% lower
## than the minimum value in that variable.
## This is done by calling the geom_miss_point() function
geom_miss_point(airquality)
```

Run (Ctrl+Enter)

(b) Description sheet

Figure 3: Overview of selected R packages, described by scope, related data and publications, as well as a small code example.

Python modules To the best of our knowledge, very few methods are already implemented for handling missing data in Python. However, one of the major libraries for machine learning and data analysis, `scikit-learn` (Pedregosa et al., 2011) has recently proposed a module for simple and multiple imputations, `sklearn.impute`. Also, as an alternative, the `statsmodels`⁸ library also has an implementation module for multiple imputation in Python now. Additionally, the `missingno` toolset (Bilogur, 2018) facilitates visualizing missing values for exploratory data analyses. We regularly survey new Python implementations for handling missing values and, if pertinent, from a theoretical and practical point of view, reference them on our platform. We expect this to promote their use but also additional assessment by practitioners and researchers from the missing values (statistics/machine learning) community.

Datasets

Especially in methodology research, an important aspect is the comparison of different methods to assess their respective strengths and weaknesses. Several datasets are recurrent in the missing values literature but have not been referenced together yet. We gathered publicly available datasets that have recurrently been used for comparison or illustration purposes in publications, R packages and tutorials. Most of these datasets are already included in R packages but some are available in other data collections. Figure 4 shows how the datasets are presented, with a detailed description shown for one of the dataset ('Ozone', obtained by clicking on its name). The description follows the UCI Machine Learning Repository presentation (Dua and Graff, 2019), including a short description of the dataset, how to obtain it, external references describing the dataset in more details, and links to tutorials/lectures on our websites or to vignettes in R packages that use the dataset.

In addition, the *Datasets* section also references existing methods for generating missing data, given assumptions on their generation mechanisms (as in the R package `mice`).

Note, however, that the list of datasets gathered here is short compared to benchmark datasets for full data methods such as the UCI Machine Learning Repository. Therefore, our proposed list also serves as an invitation to tackle this lack of a wider variety of common benchmark datasets in the missing data community.

Additional content

This unified platform collects and edits the contributions of numerous individuals who have investigated missing values problems, and developed methods to handle them. To provide an overview of some of the main actors in this field, the list of all contributors who agreed to appear on this platform is given with links to their personal or to their research lab website.

We also provide links to other interesting websites or working groups, not necessarily working with R and Python (Van Rossum and Drake, 2009) but with other programming languages such as SAS/STAT[®] and STATA (StataCorp., 2019).

Two other features are finally provided to engage the community:

1. A regularly updated list of events such as conferences or summer schools with special focus on missing values problems, and
2. A list of recurring questions together with short answers and links for more details for every question (FAQ).

3 Details of missing values workflows

After this general introduction to the 'R-miss-tastic' project and platform and the overview of its structure, we now turn to a more detailed presentation of the various workflows that we have developed and proposed on this platform.

To allow for both hands-on tutorials illustrating current practice and state-of-the-art and ready-to-use pipelines, we propose the workflows under different formats such as HTML, PDF, R Markdown (for R code) and IPython Notebook (for Python code). We encourage practitioners and researchers to use and adapt these workflows and propose modifications and improvements, in order to increase reproducibility and comparability of their work. Of course, we are aware that these workflows do not cover the entire spectrum of existing methods and data problems. The goal of the proposed workflows is rather to initiate a joint effort to create a larger spectrum of open-source workflows, and to encourage the use of standardized procedures to handle missing values. With an incomplete dataset

⁸<https://www.statsmodels.org/stable/about.html>

Incomplete data

The data sets listed below are either widely used in general in the missing data community or used for illustration of different methods handling missing values in the tutorials from the [Tutorials](#) and [R_packages](#) sections. This presentation scheme is inspired by the [UCI Machine Learning Repository](#).

Click on a table entry to obtain further information about the data set.

Name	Data Types	Attribute Types	# Instances	# Attributes	% Missing entries	Complete data available	Year
Airquality	Multivariate, Time Series	Real	154	6	7	No	1973
chorizonDL	Multivariate	Integer, Real	606	110	15	Yes	1998
Health Nutrition And Population Statistics	Multivariate, Time Series	Integer, Real	15,022	397	54	No	2017
NHANES	Multivariate	Categorical, Integer, Real	10,000	75	37	No	2012
oceanbuoys	Multivariate, Time Series	Real	736	8	3	No	1997
Ozone	Multivariate	Categorical, Integer, Real	366	13	6	No	1976
<p>Los Angeles Ozone Pollution Data, 1976. This data set contains daily measurements of ozone concentration and meteorological quantities. It can be found in R in the mlbench package and is loaded by calling <code>data(Ozone)</code>.</p> <p>More information on the dataset.</p> <p>Tutorials illustrating methods on this data:</p> <ul style="list-style-type: none"> • Julie Josse's course on missing values imputation using PC methods. • Julie Josse's and Nick Tierney's tutorial on handling missing values. Download the data set from this tutorial: ozoneNA.csv • Nick Tierney's nanIAR vignette for missing data visualization. 							
pedestrian	Multivariate, Time series	Categorical, Integer	37,700	9	2	No	2016

Figure 4: Overview of included datasets, described by key attributes and data availability.

at hand, prior to embarking on an in-depth statistical analysis, two preliminary steps are essential: (i) a descriptive analysis leveraging visualization packages such as **VIM** (Kowarik and Templ, 2016) or **naniar** (Tierney et al., 2021); (ii) a specific aim has to be defined in order to choose a specific method to use.

An example of a method whose success crucially depends on the analyst's goal is *mean imputation*: this approach is strongly counter-indicated if the aim is to estimate parameters, but it can be consistent if the aim is to predict as well as possible (Josse et al., 2019). Following this observation, our workflows are divided into different parts, defined by the objective of the statistical analysis. We aim to present and compare the main implementations available both in R and Python for each objective. Currently there are seven workflows available on the platform and we briefly present their scope and use below. For details on the implementations we encourage the reader to open the corresponding workflows, all available on the 'R-miss-tastic' platform.

How to generate missing values?

The goal of these workflows is to propose functions to generate missing values under different mechanisms. This code aims to unify classical solutions to do this. Indeed, a usual strategy to compare imputation or estimation strategies is to introduce (additional) missing values in the dataset, and use the ground truth for these missing values to evaluate the strategies (see the following section).

Rubin (1976) classifies the cause for a lack of data into three missing data mechanisms. The missing data mechanism is said to be: (i) missing completely at random (MCAR) if the lack of data is totally independent of the data values, (ii) missing at random (MAR) if the process that causes the missing data only depends on the observed values and (iii) missing not at random (MNAR) if the unavailability of the data depends on the missing variables. See Sportisse (2021) for a recent overview on the topic.

In R In the R [workflow](#)⁹, we have implemented the main function `produce_NA`¹⁰ which facilitates generating missing values under the three missing data mechanisms outlined above. This function internally calls the `ampute` function from the **mice** package (van Buuren and Groothuis-Oudshoorn, 2011) but we chose to simplify its use while adding some additional options to specify the missing values generation. In addition, the original `ampute` function generates missing values only for a complete dataset with quantitative variables¹¹. In the main function of our workflow, the user can easily introduce (additional) missing values in a complete or incomplete dataset composed of quantitative, categorical, or mixed variables, by choosing the mechanism and the percentage of missing values to be introduced. The function then returns the data matrix containing the new dataset with missing values, that also includes the missing values already present in the input data, and the indicator matrix (a binary matrix where an entry is equal to 1 if a new missing value has been generated at the same location in the data matrix and 0 otherwise).

The three main arguments are the initial dataset (`data`) in which the missing values are introduced using a given missing data mechanism (`mechanism`) and a given percentage of missing values (`perc.missing`). For example, to introduce 20% of MCAR values in the dataset X , the code is detailed below.

```
X.miss.mcar <- produce_NA(data = X, mechanism = "MCAR", perc.missing = 0.2)
X.mcar <- X.miss.mcar['data.incomp']
R.mcar <- X.miss.mcar['idx_newNA']
```

For example, if X contains three variables (fully observed) denoted as X_1 , X_2 , X_3 , two options are available to generate MAR values:

1. The first option consists in generating missing values in X_1 by using a logistic model depending on (X_2, X_3) , which are fully observed, i.e.,

$$\mathbb{P}(R_1 = 0 | X; \phi) = 1 / (1 + \exp(-(\phi_2 X_2 + \phi_3 X_3))), \quad (1)$$

where $\phi = (\phi_2, \phi_3)$ is the parameter of the missing data mechanism. In our function, ϕ is chosen such that the given percentage of missing values is achieved. This allows us to obtain missing values in the first variable X_1^{NA} . Then, the same strategy is performed to introduce missing values in X_2 and X_3 , by using a logistic model depending on (X_1, X_3) (fully observed) and

⁹<https://rmissstastic.netlify.app/how-to/generate/misssimul>

¹⁰<https://rmissstastic.netlify.app/how-to/generate/amputation.R>

¹¹If qualitative variables have previously been transformed by one-hot-encoding, they can also be handled by the `ampute` function of **mice**. The `produce_NA` function internally handles the transformation of qualitative variables prior to amputation.

(X_1, X_2) (fully observed) respectively. To get the final matrix containing missing values, we concatenate X_1^{NA} , X_2^{NA} and X_3^{NA} by handling the rows containing only missing values.

2. The second option consists in generating the missing values *by pattern*, i.e., by rows. In this case, the combinations of which variables are observed and missing are specified in a pattern matrix. For the MAR mechanism, in each pattern, at least one variable must be observed. An example (the choice by default) of such a pattern matrix is

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix},$$

where 0 indicates that the variable should have missing values whereas 1 means that it should be observed. For example, the first pattern means that the process which causes the missingness of the first variable X_1 depends on the values of X_2 and X_3 which are observed.

We also propose several ways to generate missing values, under the MNAR mechanism. It includes the most general one when the missingness depends on both the missing variables and the observed variables. It also includes the self-masked mechanism, where the unavailability of the data only depends on their values themselves. For example, it is possible to introduce self-masked missing values using a quantile censorship for which the form is specified by the argument `self.mask`, e.g., if set to 'lower', then the values are amputated based on a quantile from the lower tail of the empirical distribution such that the target proportion of missing values is achieved.

In Python To our knowledge, there is no specific module in Python to generate missing values. Consequently, we implemented such functions, in a Python workflow, which similarly to its R counterpart [workflow](#)¹² allows us to generate missing values under by different mechanisms and different percentage of missing values.¹³ The key difference with the R workflow is that the dataset must be complete and can currently only contain quantitative variables. For MAR and MNAR mechanisms, only the option *not by pattern* has been implemented. In this case, for a dataset X with three variables, a variable is chosen to be fully observed (say X_3), and the process which causes the missingness of two other variables (X_1 and X_2) depends on the values of the fully observed variable, for example with the logistic model given in (1).

How to impute missing values?

The aim of these workflows (in R and Python) is to compare the most classical imputation methods and to propose a reference pipeline for comparison on simulated and real datasets, which can be easily extended with other imputation methods. Here, the imputation methods are considered as such, i.e., the objective is not to estimate a parameter or to perform a statistical analysis on a completed dataset but to impute missing values to get a complete dataset in the best possible way. Therefore, we evaluate the methods in terms of imputation quality, by using the mean squared error (MSE). More precisely, the procedure is the following one: (i) We have access to a complete dataset X , (ii) missing values are introduced in X and we get an incomplete dataset X^{NA} , (iii) this incomplete dataset is imputed and we obtain an imputed dataset X^{imp} , (iv) the MSE, which measures the error committed by the imputation of the missing values, is computed: It is the ℓ_2 -norm of the difference of the imputed dataset and the complete one). Note that this procedure can also be performed on an incomplete dataset by introducing additional missing values. However, for now, both R and Python workflows only consider complete datasets.

Different types of imputation methods are included in this workflow:

1. imputation by the mean, which serves as a naive baseline.
2. conditional models, if the imputation relies on the conditional expectation or a draw from the conditional distribution of each variable given the others.
 - in R:
 - **mice** ([van Buuren and Groothuis-Oudshoorn, 2011](#)): a multiple imputations method by chained equations. Even if it returns several imputed datasets, they can be aggregated using the mean of the imputations to get a single imputation.
 - **missForest** ([Stekhoven and Bühlmann, 2012](#)): imputes iteratively by training random forests.

¹²https://rmissstastic.netlify.app/how-to/python/generate_html/how%20to%20generate%20missing%20values

¹³The code has been partially developed in collaboration with Boris Muzellec (Inria Paris).

- in Python:
 - `IterativeImputer` of scikit-learn library (Pedregosa et al., 2011): this function is inspired by mice, but it uses (iterative) regularized regression, imputing by the conditional expectation, and providing a simple imputation. We also use the `ExtraTreesRegressor` estimator of `IterativeImputer`, which trains iterative random forests (it is similar to `missForest` in R).
- 3. low-rank based models, the data matrix to impute is assumed to be generated as a low rank structure plus a noise term.
 - in R:
 - `softImpute` (Hastie et al., 2015): minimizes the re-weighted least squares error penalized by the nuclear norm.
 - `missMDA` (Josse et al., 2016): minimizes the re-weighted least squares error penalized by a mix between the ℓ_2 -norm and ℓ_0 -norm.
 - in Python: `softImpute` (coded for the purpose of this notebook and available [here](#)¹⁴), which minimizes the re-weighted least squares error penalized by the nuclear norm and with an internal cross-validation step to choose the regularization parameter.
- 4. machine learning methods (for the Python workflow only) using optimal transport or variational autoencoders.
 - in Python:
 - MIWAE (Mattei and Frelsen, 2019): imputes missing values with a deep latent variable model based on importance weighted variational inference.
 - Sinkhorn (Muzellec et al., 2020): randomly extracts several batches and minimizes optimal transport distances between batches to impute missing values.

For the sake of clarity, we show a comparison table (Table 1) in the Appendix, showing the difference of scope between R and Python packages used in the R-miss-tastic workflows.

In R This [workflow](#)¹⁵ provides two main functions which compares the imputation methods: (i) on a simulated dataset for different mechanisms and percentage of missing values (`how_to_impute`) or (ii) on a list of real datasets and a given mechanism and percentage of missing values (`how_to_impute_real`).

The function `how_to_impute` takes as input a complete dataset (`X`), a list of percentages of missing values (`perc.list`) and a list of missing data mechanisms (`mecha.list`). The code to use this function is given below.

```
perc.list <- c(0.1, 0.3, 0.5)
mecha.list <- c("MCAR", "MAR", "MNAR")
res <- how_to_impute(X = X, perc.list = perc.list, mecha.list = mecha.list, nbsim = 10)
```

The output of the first function `how_to_impute` is the mean of the methods' MSEs for the different missing values settings by taking the average over several repetitions (the number of repetitions can be specified through the argument `nbsim`). Figure 5 shows the output of this function and its associated plot, when the simulated dataset is Gaussian with $n = 1000$ observations, $d = 10$ variables, a mean vector such that $\mu_i = 1, \forall i \in \{1, \dots, d\}$ and a covariance matrix such that $\Sigma_{ij} = 0.5$ if $i \neq j \in \{1, \dots, d\}$, and $\Sigma_{ij} = 1$ if $i = j$. First, the mean of the methods' MSEs for the different missing values settings are reported in Figure 5a. We can note that for the MCAR mechanism, the methods perform well, while for the MNAR mechanism, the results are generally closer to those of the naive imputation by the mean. As expected, most methods give worse results for high percentages of missing values. Besides, Figure 5b shows one of the associated plot for MCAR data (there is also a plot for MAR data and a plot for MNAR data). In the first part of the Appendix, this function is illustrated for a particular dataset and the code to obtain Figure 5 is given.

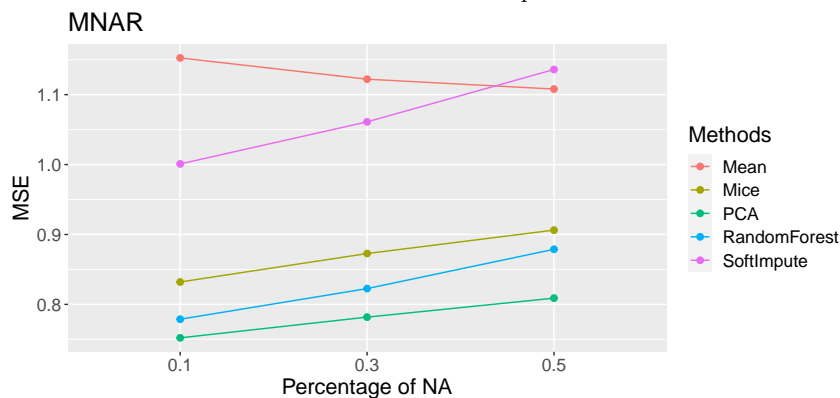
The second function `how_to_impute_real` takes as input a list of datasets (`datasets_list`), a list of missing data mechanisms (`mech`) and a given percentage of missing values (`perc`). It returns a table containing the mean of the MSEs for the simulations performed and a table for the summary plot shown in Figure 6. This can be particularly useful for practitioners who would like to have an indication of which method might be the most suited for a given or for several specific datasets. Here, the real datasets are taken from the UCI repository (Dua and Graff, 2019). An example of how to use this function in practice is detailed below.

¹⁴<https://github.com/R-miss-tastic/website/blob/master/static/how-to/python/softimpute.py>

¹⁵<https://rmissstastic.netlify.app/how-to/impute/missimp>

	0.1 MCAR	0.3 MCAR	0.5 MCAR	0.1 MAR	0.3 MAR	0.5 MAR	0.1 MNAR	0.3 MNAR	0.5 MNAR
<i>X.pca</i>	0.74	0.76	0.78	0.75	0.78	0.81	0.76	0.78	0.81
<i>X.forest</i>	0.77	0.8	0.86	0.78	0.81	0.87	0.78	0.81	0.88
<i>X.mice</i>	0.82	0.83	0.86	0.83	0.86	0.9	0.84	0.87	0.9
<i>X.soft</i>	0.93	0.86	0.87	0.97	1	1.1	1	1.1	1.1
<i>X.mean</i>	1	0.99	1	1.1	1.1	1.1	1.2	1.1	1.1

(a) Output of the function `how_to_impute` in R. The results for the MSE are truncated to two digits. Note that the line `X.pca` is the result for `missMDA`. For all methods, the default parameter choices are used.



(b) Example of plot for the MNAR mechanism.

Figure 5: Tabular and graphical outputs of the R function `how_to_impute`. The methods `mice`, `missForest`, `softImpute` and `missMDA` are compared with the naive imputation by the mean for several percentages of missing values (10%, 30%, 50%). The mean of the MSEs computed for several generations of missing values are given. In the tabular, the results are shown for several mechanisms (MCAR, MAR, MNAR) and the plot corresponds to the MNAR mechanism.

	winequality-white	winequality-red	slump	movement	decathlon
<i>X.pca</i>	0.92	0.9	0.9	0.47	0.98
<i>X.forest</i>	0.71	0.63	0.72	0.19	1
<i>X.mice</i>	0.86	0.76	0.66	0.095	1
<i>X.soft</i>	0.84	0.78	0.68	0.2	0.94
<i>X.mean</i>	1	1	1	1	1

Figure 6: Output of the R function `how_to_impute_real`. The results for the MSE are truncated to two digits. The methods `mice`, `missForest`, `softImpute` and `missMDA` for several real datasets in which 20% MCAR missing values have been introduced.

```

datasets_list <- list(wine_white = wine_white, wine_red = wine_red, slump = slump,
                    movement = movement, decathlon = decathlon)
names_dataset <- c("winequality-white", "winequality-red", "slump", "movement", "decathlon")
perc <- 0.2
mecha <- "MCAR"
res <- how_to_impute_real(datasets_list = datasets_list ,perc = perc, mech = mecha,
                        nbsim = 10, names_dataset = names_dataset)
    
```

An additional [workflow](#)¹⁶ is available and compares other deep-learning imputation strategies to most classical ones on datasets simulated either with linear relationships and nonlinear relationships. The conclusions point to better behavior of the low-rank based imputation methods even when deep-learning methods are tuned.

In Python The Python [workflow](#) is very similar to its R counterpart. The two same functions, `how_to_impute` and `how_to_impute_real`, have been implemented.

¹⁶This workflow has been implemented by an external contributor, François Husson (Professor in Statistics, France).

How to estimate parameters with missing values in R?

This R [workflow](#)¹⁷ is dedicated to a specific inferential framework when the aim is to estimate linear and logistic regression parameters for multivariate normal data. It is currently only available in R, as there are no analogous implementations available in Python to our knowledge.

There are two main methods to estimate parameters with missing values: maximum likelihood estimation adapted to missing values, using, e.g., EM-based algorithms or using multiple imputation. In this workflow, we compare two instances of these main methods, using available R implementations: the EM algorithm for logistic and linear regressions with the package `misaem` (Jiang et al., 2020) which uses the Stochastic Approximation of EM algorithm (SAEM Delyon et al., 1999) and multiple imputation with the package `mice`. Both strategies are valid under the MAR missing data mechanism. The workflow performs the estimation on a simulated dataset, but the dataset can be replaced with any custom dataset that the user believes satisfies the assumptions about the missing data mechanism and distribution of covariates.

The `misaem` package facilitates estimation of parameters of linear and logistic regression models from incomplete data, and also provides valid estimates of these parameters' variances. The functions `miss.lm`, `miss.glm` resemble the standard `lm` and `glm` functions both in terms of their signature and output.

The rationale behind the popular multiple imputation approach is to create $M > 1$ complete datasets by imputing the missing values with plausible values, and then to estimate a parameter of interest θ on each of the imputed datasets. The multiple estimations of θ and their variability reflect the uncertainty due to the unknown missing values. The parameter estimation is performed by applying the analytic method used, had the data been complete. This provides an estimate of the parameter θ and an estimate of the corresponding variance, for each imputed dataset. These quantities are finally 'pooled' by using specific rules named "Rubin's rules" (Rubin, 2004), leading to a final point estimate, with a corresponding estimation of its variance that takes into account the uncertainty due to missing values.

In the corresponding workflow, we compare this method to the previous EM algorithm and provide the basic lines of code required to estimate parameters of linear or logistic regression models with incomplete covariate data.

For an additional example of how to estimate regression parameters, we refer to the [tutorial](#)¹⁸ on handling missing values in R by Julie Josse: it walks through a complete analysis, covering visualization of missing data patterns, data visualization, dimensionality reduction of incomplete data, and regression, in the presence of missing data.

How to predict in the presence of missing values?

As mentioned in the introduction, methods to deal with missing values are not the same when the aim is to estimate parameters or to predict a target variable. Josse et al. (2019) study the problem of supervised learning with missing values, i.e., when the aim is to predict an outcome y , from incomplete covariates in X . Note that contrary to the estimation setting, supervised learning involves training and test sets and both may have missing values. Josse et al. (2019) recommend to impute the training set and the test set with a same constant, such as the mean, and then to apply a universally consistent learner, i.e., a very powerful learner, such as gradient boosting, able to learn or fit any function. When forest-based methods are used to do prediction, another method is available, the Missing Incorporated in Attributes (MIA) method (Twala et al., 2008). Note that constant imputation or MIA are recommended asymptotically but when having limited data in the prediction setting, other imputation methods can outperform these asymptotically consistent methods (Josse et al., 2019). This is explored in the following workflows. The different methods are compared in terms of quality of the prediction of the outcome (AUC for a binary outcome and MSE for a continuous outcome).

In R The R [workflow](#)¹⁹ assesses a popular strategy (two-step strategy) which involves independently imputing the training and test sets using the same imputation method. These datasets are then treated as being complete data, and regular learning algorithms are applied to predict some target variable.²⁰ Several imputation methods are compared, such as `mice`, `missForest`, `softImpute`, and mean imputation. Note that, until recently, using the popular `mice` package for learning predictive

¹⁷<https://rmissstastic.netlify.app/how-to/estimate/missestim>

¹⁸https://rmissstastic.netlify.app/tutorials/josse_bookdown_dataanalysismissingr_2020

¹⁹https://rmissstastic.netlify.app/how-to/external/how_to_predict_in_r

²⁰This workflow has been written by an external contributor of the website, Katarzyna Woźnica (PhD student at the Warsaw University of Technology, Poland).

models on incomplete data in R was hindered by the fact that it did not allow using the same imputation model for the training and test set. This has, however, been addressed with the argument `ignore` of the R function `mice`, the details of this recent extension can be found on GitHub.²¹

In Python The Python [workflow](#)²² compares two strategies, where the aim is to predict a target variable and the covariates may contain missing values:

1. The *two-step* strategy consists of imputing the missing values both in the training and in the test set with a method like mean imputation or `IterativeImputer` of the `scikit-learn` library, and to apply usual learning algorithms (random forests, gradient boosting, linear regression) on the imputed dataset. This learning algorithm can be applied to the imputed dataset \tilde{X} but also to a new variable made of the combination of the imputed covariates \tilde{X} with the response pattern R : $[\tilde{X}, R]$.
2. The *one-step* strategy performs prediction using learning methods adapted to the missing data without necessarily imputing them, such as the MIA method (Twala et al., 2008), which is in our notebook.

We propose a function, `score_pred`, which compares these strategies in terms of prediction performances by introducing missing values in complete covariates (`x_comp`) under a specific missing data mechanism (`mecha`) and a given percentage of missing values (`p`). The code for calling this function is given below, when the learning algorithm is the gradient boosting and 20% of MCAR values are introduced.

```
learner = HistGradientBoostingRegressor()
p = 0.2
res = score_pred(x_comp=X, y = y, learner=learner , p=p, nbsim=10, mecha="MCAR")
```

The dataset is then split into a training set and a test set (75% in the training set, 25% in the test set) and the methods presented below are applied by considering a specific learning algorithm. The function then returns the prediction error on the test set, by comparing the ground truth (`y`) and the predicted outcome values on the test set for each simulation (i.e., each run for the generation of missing values). Figure 7 shows the graphical output of this function called for different learning algorithms (linear regression, random forests and gradient boosting) and for different missing data mechanisms (20% MCAR and MNAR, see the section on how to generate missing values). When the learner is linear regression, the two-step methods with added mask, both for the MCAR mechanism and the MNAR mechanism, perform well. Since the simulated dataset is generated using a linear regression model, the linear regression is expected to give better results than the other learners. In addition, for the MNAR mechanism, the one-step strategy *MIA* (especially when the gradient boosting is performed) appears to be a good choice.

Another function is specifically designed to handle datasets which already contain missing values. The second part of the Appendix shows a concrete example of this notebook on a real dataset.

This concludes the overview of the workflows developed in this project. We invite other practitioners and researchers to use and extend these methods. Overall, we hope that by creating and sharing implemented methods, new methods can be more easily developed and easily compared and evaluated.

4 Perspectives and future extensions

By providing a platform and community to discuss missing data, software, approaches and workflows, the sharing of expertise on missing data can hopefully be improved and extended more easily.

Towards uniformization and reproducibility

One way to promote and encourage practitioners and researchers in their work with missing values is to provide benchmarks and workflows around missing data. As has been shown in data competitions, community involvement produces many creative solutions and discussions that move the field forward, and challenge existing strategies. We will continue to work on our workflows and related source code. In doing so, we hope to encourage users to continue to test new methods and present results in a clear and reproducible manner. In addition, we plan to propose two types of data

²¹<https://github.com/amices/mice/issues/32>

²²https://rmisstastic.netlify.app/how-to/python/predict_html/how%20to%20predict

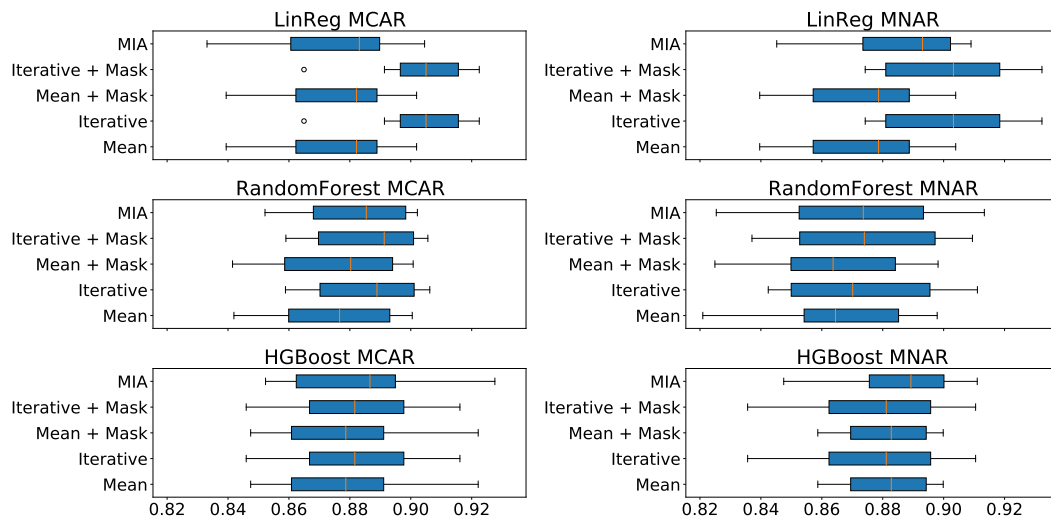


Figure 7: Plot of the function `score_pred` to compare different strategies when the aim is to predict in Python. 20% of missing values are introduced in a simulated dataset using the MCAR mechanism or the MNAR mechanism. The covariates $X \in \mathbb{R}^{1000 \times 3}$ are generated under a multivariate Gaussian distribution, the parameter of the regression $\beta \in \mathbb{R}^3$ follows a random uniform distribution. The outcome y is generated according to a linear model such that $y = X\beta + \epsilon$, with ϵ representing Gaussian noise. The two-step strategies (IterativeImputer and the mean imputation) with or without adding a mask and the one-step strategy MIA are compared in terms of prediction error, and several learners are performed (linear regression, random forests, gradient boosting). The closer the result is to 1, the more accurate the prediction is (1 corresponds to perfect prediction, 0 to the worst prediction).

challenges: 1) imputation and estimation, and 2) analysis workflows. For the first part of the challenge, the objective is to find the best imputation or estimation strategy. The community will be given a dataset with missing values, for which there is actually a hidden copy of the real values. The community will then get the task of creating imputed values, which are assessed against the original dataset with complete values, to determine which imputation is best. This is similar in spirit to the Netflix prize (Bennett et al., 2007) and the M4 challenge in the time series domain (Makridakis et al., 2018). This benchmarking could be extended to other areas, such as parameter estimation, and predictive modeling with missing data. Analysis workflows could form another community challenge, assessed in a similar way to existing ‘datathon’ events where entries are assessed by an expert panel. Here the challenge could be to develop workflows and data visualizations from complex data. The data could have challenging features, and be combined from various data sources with complex structure, such as data with several types of missingness, images, text data, longitudinal data, and time series.

Future extensions

Possible enhancements that could be added in future releases of the platform, for which we welcome suggestions and contributions, are the following: A workflow with a focus on MNAR data and different solutions that can handle such data (as diversity of existing solutions is large, such a unified workflow will be a consequential contribution); for more applied users, a comparison of computation times of different methods, benchmarked on various types of data. Another problem that is becoming more common is missing values in data integration. Indeed, questions such as *what do I do when I have clinical data from multiple centers with different mechanisms of missing values or with systematically missing values in certain data?* or *what do I do when I have time series and missing values in one of the groups of variables?* would be also worth addressing in additional workflows.

Participation and interaction

This platform is aimed to offer a venue for the community, in the sense that we welcome every comment and question, encourage submissions of new works, theoretical or practical, either through the provided contact form or directly via the GitHub project repository. We have already received useful feedback and several external contributions, organized several remote calls and working sessions at statistics conferences. We are planning on regularly relaunching calls for new material for

the platform, for example through the R consortium blog²³, R-bloggers²⁴ and social media platforms. We also intend to use these channels to communicate more generally about the platform and the topic of missing values.

In order for the platform to be a reference to the community, it must provide regularly updated, user-friendly content. To achieve this goal, it is important to propose sustainable and accessible solutions for the maintenance of the 'R-miss-tastic' platform. We hope that the well documented source code of the platform facilitates external contributions and community feedback on this project.

In conclusion, the aim of this platform is to go beyond mere community participation, namely to seed meaningful community interactions, and to offer a hub of communication among groups that rarely exchange, both within, and between academia and industry.

5 Acknowledgements

This work has partially been funded by the R Consortium, Inc. We would like to thank Steffen MORITZ and François HUSSON for their active support and feedback, all contributors who have generously made their course and tutorial materials available, as well as the contributors to the workflows in R and Python code.

Bibliography

- J. Bennett, S. Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer, 2007. [p258]
- F. Biessmann, D. Salinas, S. Schelter, P. Schmidt, and D. Lange. "deep" learning for missing value imputation in tables with non-numerical data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM '18*, pages 2017–2025, 2018. ISBN 978-1-4503-6014-2. doi: 10.1145/3269206.3272005. URL <http://doi.acm.org/10.1145/3269206.3272005>. [p245]
- A. Bilogur. Missingno: a missing data visualization suite. *Journal of Open Source Software*, 3(22):547, 2018. doi: 10.21105/joss.00547. URL <https://doi.org/10.21105/joss.00547>. [p250]
- L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. [p246]
- J. Carpenter and M. Kenward. *Multiple Imputation and its Application*. John Wiley & Sons, Dec. 2012. [p244, 247]
- X. Cheng, D. Cook, and H. Hofmann. Visually exploring missing values in multivariable data using a graphical user interface. *Journal of statistical software*, 68(1):1–23, 2015. [p244]
- B. Delyon, M. Lavielle, E. Moulines, et al. Convergence of a stochastic approximation version of the em algorithm. *The Annals of Statistics*, 27(1):94–128, 1999. [p256]
- D. Dua and C. Graff. UCI machine learning repository, 2019. URL <http://archive.ics.uci.edu/ml>. [p250, 254]
- C. K. Enders. *Applied missing data analysis*. Guilford press, 2010. [p244]
- A. Gelman and J. Hill. Opening windows to the black box. *Journal of Statistical Software*, 40, 2011. [p244]
- L. Gondara and K. Wang. Mida: Multiple imputation using denoising autoencoders. In D. Phung, V. Tseng, G. Webb, B. Ho, M. Ganji, and L. Rashidi, editors, *Proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2018)*, Lecture Notes in Computer Science, pages 260–272. Springer International Publishing, 2018. ISBN 3319930404. doi: 10.1007/978-3-319-93040-4_21. URL <https://arxiv.org/abs/1705.02737>. [p247]
- T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh. Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402, 2015. [p254]
- J. Honaker, G. King, and M. Blackwell. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47, 2011. URL <http://www.jstatsoft.org/v45/i07/>. [p244]

²³<https://www.r-consortium.org/news/blog>

²⁴<https://www.r-bloggers.com/>

- R. Ihaka. R: Past and future history. *Computing Science and Statistics*, 392396, 1998. [p244]
- W. Jiang, J. Josse, M. Lavielle, and T. Group. Logistic regression with missing covariates—parameter estimation, model selection and prediction within a joint-modeling framework. *Computational Statistics & Data Analysis*, 145:106907, 2020. [p256]
- J. Josse and J. P. Reiter. Introduction to the special section on missing data. *Statistical Science*, 33(2): 139–141, 2018. [p244]
- J. Josse, F. Husson, et al. *missmda*: a package for handling missing values in multivariate data analysis. *Journal of Statistical Software*, 70(1):1–31, 2016. [p244, 254]
- J. Josse, N. Prost, E. Scornet, and G. Varoquaux. On the consistency of supervised learning with missing values. *arXiv preprint*, 2019. URL <https://arxiv.org/abs/1902.06931>. [p247, 252, 256]
- A. Kowarik and M. Templ. Imputation with the R package VIM. *Journal of Statistical Software*, 74(7): 1–16, 2016. doi: 10.18637/jss.v074.i07. [p244, 252]
- S. Lê, J. Josse, and F. Husson. Factominer: an r package for multivariate analysis. *Journal of statistical software*, 25:1–18, 2008. [p262]
- R. J. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019. [p244, 247]
- S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018. [p258]
- P.-A. Mattei and J. Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *International Conference on Machine Learning*, pages 4413–4423. PMLR, 2019. [p254]
- I. Mayer, E. Sverdrup, T. Gauss, J.-D. Moyer, S. Wager, and J. Josse. Doubly robust treatment effect estimation with missing attributes. *Ann. Appl. Statist.*, 14(3):1409–1431, 2020. ISSN 1932-6157. doi: 10.1214/20-AOAS1356. [p246, 247]
- S. Moritz and T. Bartz-Beielstein. *imputeTS*: Time Series Missing Value Imputation in R. *The R Journal*, 9(1):207–218, 2017. doi: 10.32614/RJ-2017-009. [p246]
- J. S. Murray and J. P. Reiter. Multiple imputation of missing categorical and continuous values via bayesian mixture models with local dependence, 2015. URL <http://arxiv.org/abs/1410.0438>. [p244]
- B. Muzellec, J. Josse, C. Boyer, and M. Cuturi. Missing data imputation using optimal transport. In *International Conference on Machine Learning*, pages 7130–7140. PMLR, 2020. [p254]
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. [p245, 250, 254]
- D. B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976. [p252]
- D. B. Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004. [p256]
- J. L. Schafer and J. W. Graham. Missing data: our view of the state of the art. *Psychological methods*, 7(2):147–177, June 2002. [p244, 247]
- A. Sportisse. Handling heterogeneous and mnar missing data in statistical learning frameworks: imputation based on low-rank models, online linear regression with sgd, and model-based clustering. 2021. [p252]
- StataCorp. *Stata Statistical Software: Release 16*. StataCorp LLC., College Station, TX, 2019. [p250]
- D. J. Stekhoven and P. Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012. [p253]
- N. Tierney, D. Cook, M. McBain, and C. Fay. *naniar: Data Structures, Summaries, and Visualisations for Missing Data*, 2021. URL <https://CRAN.R-project.org/package=naniar>. R package version 0.6.1. [p244, 252]
- N. J. Tierney and D. H. Cook. Expanding tidy data principles to facilitate missing data exploration, visualization and assessment of imputations. *arXiv preprint arXiv:1809.02264*, 2018. [p244]

- B. Twala, M. Jones, and D. J. Hand. Good methods for coping with missing data in decision trees. *Pattern Recognition Letters*, 29(7):950–956, 2008. [p256, 257]
- S. van Buuren. *Flexible Imputation of Missing Data, Second Edition*. CRC Press, 2018. [p244, 247]
- S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67, 2011. URL <http://www.jstatsoft.org/v45/i03/>. [p244, 252, 253]
- G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697. [p244, 250]
- Y. Xie, A. Presmanes Hill, and A. Thomas. *blogdown: Creating Websites with R Markdown*. The R Series. Chapman and Hall/CRC, 2017. ISBN 978-0815363729. [p245]
- Z. Zhu, T. Wang, and R. J. Samworth. High-dimensional principal component analysis with heterogeneous missingness. *arXiv preprint arXiv:1906.12125*, 2019. [p244]

1 Appendix

Tutorial for imputing missing values in R

The goal of this tutorial is to give practical details on the [R-workflow](#) entitled *How to impute missing values?*²⁵. In this workflow, users can compare the most popular methods to impute missing values in R on simulated or real datasets.

We illustrate this workflow by considering a small dataset called *decathlon*, that contains athletes' performance during two sporting events (41 rows, 13 columns²⁶). It is available in the R package **FactoMineR** (Lê et al., 2008).

```
library(FactoMineR)
data(decathlon)
head(decathlon[,1:4]) # four first columns of the dataset
      100m Long.jump Shot.put High.jump
SEBRLE 11.04      7.58   14.83    2.07
CLAY    10.76      7.40   14.26    1.86
KARPOV  11.02      7.30   14.77    2.04
BERNARD 11.02      7.23   14.25    1.92
YURKOV  11.34      7.09   15.19    2.10
WARNERS 11.11      7.60   14.31    1.98
```

If we have collected similar data, e.g., described by the same variables but for new athletes, that contain missing values, practitioners may want to know how to impute such a dataset. To address this question, we can introduce missing values under different mechanisms (MCAR, MAR or MNAR) and with different percentages of missing values (here we compare 20% and 50%) in the complete dataset and compare some imputation methods in terms of mean squared error (MSE), i.e., the error committed by the imputation of the missing values. Missing values are introduced in all covariates. The function `how_to_impute` can be used to compare the imputation methods described in the section on how to impute missing values (**missMDA**, **mice**, **missForest**, **softImpute** and the imputation by the mean) using different percentages and types of missing values given in two lists by the users. More particularly, the arguments are the following ones: the complete dataset where the missing values will be introduced (`X`), a list containing the different percentage of missing values (`perc.list`), a list containing the different missing-data mechanisms (`mecha.list`) and the number of simulations performed (`nbsim`). Note that for **missMDA**, the number of components in the PCA used to predict the missing entries is estimated using a cross-validation with the function `estim_ncpPCA`. For **softImpute**, we use a cross-validation to choose the regularization parameter (coded for the purpose of the notebook). This function returns a table with the mean of the MSEs over the simulations for the different methods and for the different missing data settings (20% MCAR values, 50% MCAR values, 20% MAR values, 50% MAR values, 20% MNAR values, 50% MNAR values).

```
perc.list <- c(0.2,0.5)
mecha.list <- c("MCAR", "MAR", "MNAR")
res <- how_to_impute(X=decat_sc, perc.list=perc.list,mecha.list=mecha.list,nbsim=10)
res
```

	0.2 MCAR	0.5 MCAR	0.2 MAR	0.5 MAR	0.2 MNAR	0.5 MNAR
X.pca	0.8822782	1.0537611	0.9394561	1.0873315	0.9876867	1.1026891
X.forest	0.8820789	0.9577351	0.9403659	1.0526915	0.9940827	1.0809478
X.mice	0.8610320	1.0372518	0.9559042	1.0948981	0.9887239	1.1271581
X.soft	0.7935545	0.8865989	0.8721907	0.9556373	0.8951239	0.9692859
X.mean	1.0177192	1.0306089	1.0972080	1.0770715	1.1342289	1.1077467

With this result in hand, we can easily visualize some of the results. Figure 8 shows the associated graphics, for each of the missing-data mechanism. The code to obtain these graphics is given below.

```
plotdf <- do.call(c, res)
plotdf <- as.data.frame(plotdf)
names(plotdf) <- 'mse'
n_perc.list <- length(perc.list)
n_mecha.list <- length(mecha.list)
```

²⁵This tutorial is only an example of use but more practical details are given in the original workflow.

²⁶We do not consider the last variable in this part, which is categorical. Some imputation methods do not handle mixed data.

```

methods.list <- c("PCA", "RandomForest", "Mice", "SoftImpute", "Mean")
meth <- rep(methods.list, n_perc.list * n_mecha.list)
plotdf <- cbind(plotdf, meth)
perc <- rep(rep(as.character(perc.list), each = 5), length(mecha.list))
plotdf <- cbind(plotdf, perc)
mecha <- rep(mecha.list, each = 5 * length(perc.list))
plotdf <- cbind(plotdf, mecha)

# For MCAR data
ggplot(plotdf[plotdf['mecha'] == "MCAR",])
+ geom_point(aes(x = perc, y = mse, color = meth), size = 2)
+ ylab("MSE") + xlab("Percentage of NA")
+ geom_path(aes(x = perc, y = mse, color = meth, group = meth))
+ ggtitle("MNAR") + labs(color = "Methods") + theme(text = element_text(size = 20))

# For MAR data
ggplot(plotdf[plotdf['mecha'] == "MCAR",])
+ geom_point(aes(x = perc, y = mse, color = meth), size = 2)
+ ylab("MSE") + xlab("Percentage of NA")
+ geom_path(aes(x = perc, y = mse, color = meth, group = meth))
+ ggtitle("MNAR") + labs(color = "Methods") + theme(text = element_text(size = 20))

# For MNAR data
ggplot(plotdf[plotdf['mecha'] == "MCAR",])
+ geom_point(aes(x = perc, y = mse, color = meth), size = 2)
+ ylab("MSE") + xlab("Percentage of NA")
+ geom_path(aes(x = perc, y = mse, color = meth, group = meth))
+ ggtitle("MNAR") + labs(color = "Methods") + theme(text = element_text(size = 20))

```

For this dataset and these missing data settings, **softImpute** appears to be the best imputation method.

Tutorial for predicting in presence of missing values in Python

The goal of this tutorial is to give practical details on the [Python-workflow](#) entitled *How to predict with missing values?*²⁷. In this workflow, users can compare methods in Python to predict a target variable when the covariates contain missing values.

We consider the dataset called *california_housing* (20640 rows, 9 columns). The target variable is the median house value for California districts and the covariates provide information (latitude, longitude, number of people in the district...) on the different districts.

If we know that new observations will contain missing values, an interesting question is how to predict the target variable in presence of covariates with missing values. To answer this, we can impute missing values in the covariates and compare methods which handle them and predict the target variable.

First, we generate missing values in the covariates of the dataset *california_housing*, using the function `produce_NA`. The three main arguments are the initial dataset (X) in which missing values are introduced using a given missing data mechanism (*mecha*) and a given percentage of missing values (*p_miss*). In the following example, we introduce 20% MCAR values.

```

XproduceNA_MCAR = produce_NA(X = x_comp, p_miss = 0.2, mecha = "MCAR")
x_MCAR = XproduceNA_MCAR['X_incomp'].numpy()

```

To predict, we consider two strategies presented in the section on how to predict in the presence of missing values: (i) The *two-step* strategy which consists of imputing missing values and applying classical methods on the completed datasets to predict, and (ii) the *one-step* strategy which predicts using methods adapted to the missing values without necessarily imputing them. The code below allows comparison of different prediction strategies nested in a two-step or a one-step strategy. To do so, we use the function `plot_score_realdatasets`, which handles datasets already containing missing values. Figure 9 shows the graphical output of this function called for different learning algorithms. When the learner is the linear regression, the two-step methods with added mask perform

²⁷This tutorial is only an example of use but more practical details are given in the original workflow.

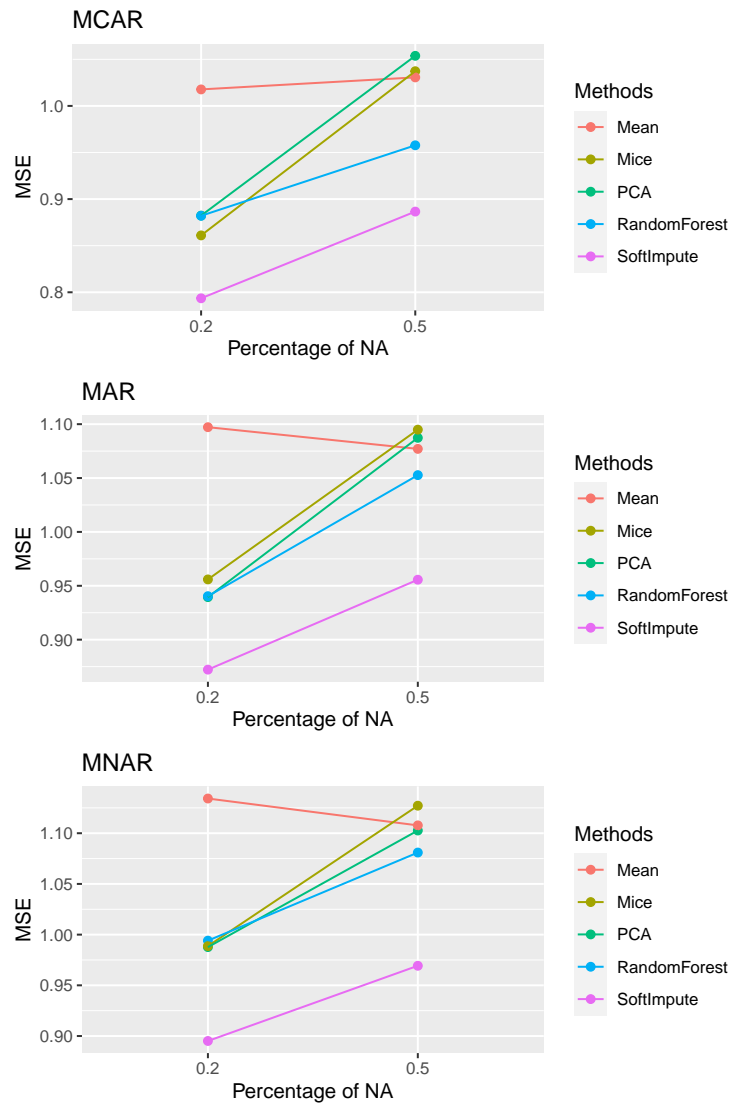


Figure 8: Graphical outputs of the R function `how_to_impute`. The methods **mice**, **missForest**, **softImpute** and **missMDA** are compared with the naive imputation by the mean for several percentages of missing values (20%, 50%). The mean of the MSEs computed for several generations of missing values are given. The results are shown for different mechanisms (MCAR, MAR, MNAR).

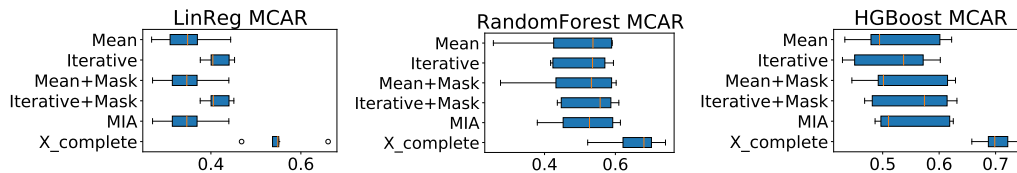


Figure 9: Graphical outputs of the Python function `plot_score_realdatasets`. The x-axis indicates the MSE. The two-steps methods considering the imputation by the mean (Mean), IterativeImputer (Iterative) with or without adding the mask and the one-step method MIA are compared with the case without missing values (X_complete). Note that the mask is the binary matrix which indicates where are the missing values. When we add the mask, we consider then an augmented matrix, with the initial matrix and the mask. The learner are the linear regression (left panel), the random forests (middle panel) and gradient boosting (right panel).

well. Indeed, since the simulated dataset is generated considering a linear regression, the linear regression is expected to give better results than the other learners.

The code for the function `plot_score_realdatasets` is given below. The main arguments are the dataset (X), the outcome variable (y) and the learning algorithm to use (learner).

```
learners = {'LinReg': LinearRegression(),
           'RandomForest': RandomForestRegressor(),
           'HGBBoost': HistGradientBoostingRegressor()}
for learner_name, learner in learners.items():
    plt.figure(figsize=(10,10))
    for ii, (X, X_name) in enumerate(zip([x_MCAR], ['MCAR'])):
        plt.subplot(3, 2, ii+1)
        plot_score_realdatasets(X, y, learner, learner_name + ' ' + X_name, x_comp)
```

Table 1: Comparison of the scope and functionality of different R and Python packages used for the same tasks in the R-miss-tastic R and Python workflows. If existing, differences between approximately equivalent packages or functions are summarized in the last column.

R implementation	Scope	Python counterpart	Differences (if any)
imputeMean (implemented in R-miss-tastic workflow)	Impute missing values of each variable by their means.	module sklearn.impute, SimpleImputer with strategy='mean'	
softImpute	Impute missing values using a low-rank completion with nuclear norm penalties	function softImpute (implemented in R-miss-tastic workflow)	The R package is better optimized than our Python version. The R implementation also has different optimization algorithms implemented (iterative SVD, iterative ALS).
mice	Give multivariate imputations by chained equations	module sklearn.impute IterativeImputer with BayesianRidge	The Python module uses iterative chained equations. However, it differs from the mice package, because it uses a ridge iterate and it returns by default a single imputation. Note that the argument <code>sample_posterior=True</code> allows to get stochastic imputations, and not multiple imputations, as the R function mice does.
missForest	Impute missing values using random forests	module sklearn.impute, IterativeImputer with ExtraTreesRegressor	The Python implementation does not exactly use random forests with CART trees, but forests with trees which choose a random split (instead of the best split per feature).
missMDA	Impute missing values using a low-rank matrix completion with penalty		

Imke Mayer
Institute of Public Health, Charité – Universitätsmedizin Berlin
imke.mayer@charite.de

Aude Sportisse
Maasai, Inria Sophia Antipolis
aude.sportisse@inria.fr

Julie Josse
PreMeDICaL, Inria Sophia Antipolis
julie.josse@inria.fr

Nicholas Tierney
Department of Econometrics and Business Statistics, Monash University
nicholas.tierney@gmail.com

Nathalie Vialaneix
MIAT, Université de Toulouse, INRA
nathalie.vialaneix@inrae.fr

An Open-Source Implementation of the CMPS Algorithm for Assessing Similarity of Bullets

by Wangqian Ju and Heike Hofmann

Abstract In this paper, we introduce the R package **cmpsR**, an open-source implementation of the Congruent Matching Profile Segments (CMPS) method developed at the National Institute of Standards and Technology (NIST) for objective comparison of striated tool marks. The functionality of the package is showcased by examples of bullet signatures that come with the package. Graphing tools are implemented in the package as well for users to assess and understand the CMPS results. Initial tests were performed on bullet signatures generated from two sets of 3D scans in the Hamby study under the framework suggested by the R package `bulletxtrctr`. New metrics based on CMPS scores are introduced and compared with existing metrics. A measure called sum of squares ratio is included, and how it can be used for evaluating different scans, metrics, or parameters is showcased with the Hamby study data sets. An open-source implementation of the CMPS algorithm makes the algorithm more accessible, generates reproducible results, and facilitates further studies of the algorithm such as method comparisons.

Introduction

In this paper, we present an open-source implementation of the algorithm of the Congruent Matching Profile Segments (CMPS) method. Chen et al. (2019) developed the CMPS method for “objective comparison of striated tool marks” and demonstrated its use in some examples of comparing bullet signature correlations. Although Chen et al. (2019) conceptually describe the CMPS algorithm in their paper, the authors did not release an implementation of their method. Thus, our effort here is to introduce the CMPS method to the R community and provide an open-source, publicly available implementation of the algorithm to use, review, and improve. Our implementation is made available as part of the R package **cmpsR** on CRAN.

According to the Uniform Crime Reporting Program of the FBI (Federal Bureau of Investigation), “more than 76 percent (76.7) of the homicides for which the FBI received weapons data in 2020 involved the use of firearms” (United States Department of Justice, Federal Bureau of Investigation., n.d.a). At the same time, the number of murder victims jumped between 2019 and 2020 by more than 23 percent (23.4) to 17754 (United States Department of Justice, Federal Bureau of Investigation., n.d.b). This increase is unprecedented and highlights the important role that firearm examination plays in all these cases. An important task of firearm examination is to answer the question of whether two pieces of evidence come from the same source or whether a piece of evidence matches a sample obtained from a specific firearm. Here, in particular, we are interested to determine whether two bullets were fired from the same gun barrel. Assessing the similarity between two bullets is based on a comparison of striation marks acquired during the firing process as bullets are propelled through the barrel. The current state of the art sees firearms examiners make an assessment of similarity based on a visual comparison, generally, using a comparison microscope (AFTE Criteria for Identification Committee 1992). This practice has been criticized for its lack of objectivity and the associated problem of determining valid error rates (President’s Council of Advisors on Science and Technology 2016).

A report published by the Committee on Identifying the Needs of the Forensic Sciences of the National Research Council (2009) states that “[m]uch forensic evidence—including, for example, bite marks and firearm and toolmark identification—is introduced in criminal trials without any meaningful scientific validation, determination of error rates, or reliability testing to explain the limits of the discipline.” To overcome those criticisms and concerns, researchers have been making an effort to build databases and develop frameworks and algorithms that bring an objective and quantitative assessment into the field. The database of Zheng (2016) with digital 3D topographic scans from various studies by Brundage (1998), Hamby, Brundage, and Thorpe (2009), Hamby et al. (2019) and others, provides a great resource for researchers. Algorithms that can be validated and tested and generate quantitative results are developed for 2D and 3D surface texture (Song et al. 2005), tool marks (Chumbley et al. 2010), and striae on Land Engraved Areas (LEAs) of bullets Chen et al. (2019). The notion of bullet signatures is one of the results of these efforts, and we will use it to explain the CMPS algorithm and showcase the **cmpsR** implementation. Bullet signatures can be extracted from bullet LEAs, typically one for each land engraved area, and how those bullet signatures are extracted from scans will be discussed in the Background section. Bullet signatures capture striation marks on the bullet in a numeric format and therefore serve as the foundation for algorithms. These scans played

an important role in bringing objectivity into the field. In the following sections, we will: review the background of bullet signature comparisons, discuss how we followed the idea described by Chen et al. (2019) for the implementation of the CMPS algorithm, propose new metrics based on the CMPS score and a principled evaluation framework for algorithmic results comparison, and present results of applying our implementation to real data.

Background

Hamby data set

The datasets we worked with come from the James Hamby Consecutively Rifled Ruger Barrel Study (Brundage 1998; Hamby, Brundage, and Thorpe 2009; Hamby et al. 2019), in particular, Hamby set 252 and Hamby set 44. For each Hamby set, a total of 35 bullets is fired from (the same) ten consecutively manufactured Ruger P-85 pistol barrels. Two bullets are fired from each barrel, making up a set of 20 reference bullets. An additional 15 bullets are fired from these ten barrels in a fashion unknown to the study participant. The aim of the Hamby Study was to have firearms examiners identify which barrel each of the 15 questioned bullets was fired from. The Ruger P-85 barrels are traditionally rifled barrels with six grooves and lands as shown in Figure 1. During the firing process, grooves and lands are engraved on a bullet. Firearms examiners use striation marks on land engraved areas (LEAs) for their visual comparison. For algorithmic purposes, 3D topographical images of land engraved areas were obtained and stored in x3p format (XML 3-D Surface Profile). The x3p format provides a standard way of exchanging 2D and 3D profile data. It conforms to the [ISO5436-2 standard](#) adopted by the OpenFMC (Open Forensic Metrology Consortium), a group of firearm forensics researchers who contributes to the establishment of best practices of using metrology in forensic science. Hamby set 252 was scanned using a NanoFocus lens at 20x magnification with the scan resolution being $1.5625 \mu\text{m} \times 1.5625 \mu\text{m}$ per pixel. Hamby set 44 was scanned at the Roy J Carver High-Resolution microscopy lab at Iowa State. These scans were acquired with a Sensofar Confocal Light Microscope at 20x magnification for a nominal resolution of $0.645 \mu\text{m} \times 0.645 \mu\text{m}$ per pixel. Both Hamby set 252 and Hamby set 44 are publicly available from the NIST Ballistics Database Project (Zheng 2016).

Extracting signatures from LEA scans

The automated framework for extracting signatures from x3p files used in this paper was proposed by Hare, Hofmann, and Carriquiry (2017) and is implemented in the R packages `x3ptools` (Hofmann et al. 2020) and `bulletxctrctr` (Hofmann, Vanderplas, and Krishnan 2019). `x3ptools` is a package to read, write, and generally, process x3p files. The `bulletxctrctr` package implements a pipeline for extracting and comparing signatures from scans of land-engraved areas.

Figure 2 gives an overview of all of the steps in the process from scan to signatures. Figure 2(a) shows a rendering of a 3D scan of a bullet land engraved area. The raised portion of the surface on the left and right of the scan are parts of the adjacent groove engraved areas (GEAs), the middle area shows a land engraved area with well expressed striation marks. The first step of obtaining the bullet signature is to extract a cross-section at a fixed height on the land engraving.

The thin white horizontal line in Figure 2(a) indicates which cross-section was identified by the algorithm to represent the LEA; Figure 2(b) shows the corresponding cross-sectional view. Groove engraved areas are removed from the analysis as indicated by the vertical blue lines in Figure 2(c) and Figure 2(d). A non-parametric LOESS smooth (Cleveland, Grosse, and Shyu 1991) is fitted to capture the bullet curvature (Figure 2(e)) and, finally, the **bullet signature** (Figure 2(f)) is obtained as residuals of the cross-section and the fitted smooth. Note that in Chen et al. (2019) bullet signatures are referred to as bullet profiles. However, to avoid confusion, we distinguish the notion of bullet signatures from bullet profiles. Bullet profiles are shown in panels (b), (c), and (d) of Figure 2, while Figure 2(f) shows the corresponding bullet signature. Identifying the groove engraved areas correctly is fundamental for a correct downstream analysis of the signatures. We have provided an interactive web application to allow for a human-in-the-middle inspection and intervention. It is implemented as an R Shiny App (Chang et al. 2021) named `bulletinspectR` for identifying and correcting those errors. An example of the extraction process with corresponding code and parameter settings can be found in the “Supplementary materials”. Note that the process of extracting signatures might be different from the one used in Chen et al. (2019) because no code or parameter settings are made available publicly.

Conceptual idea of CMPS

Most algorithms for comparing striation marks are based on the digitized signatures and produce a similarity score Krishnan and Hofmann (2019). The congruent matching profile segments (CMPS) algorithm, developed by Chen et al. (2019) for “objective comparison of striated tool marks”, is one such algorithm. The algorithm’s main idea is to take a set of consecutive and non-overlapping basis segments from the comparison and for each segment find the “best” registration position on the reference (the other bullet signature) with respect to their cross-correlation values. From a comparison

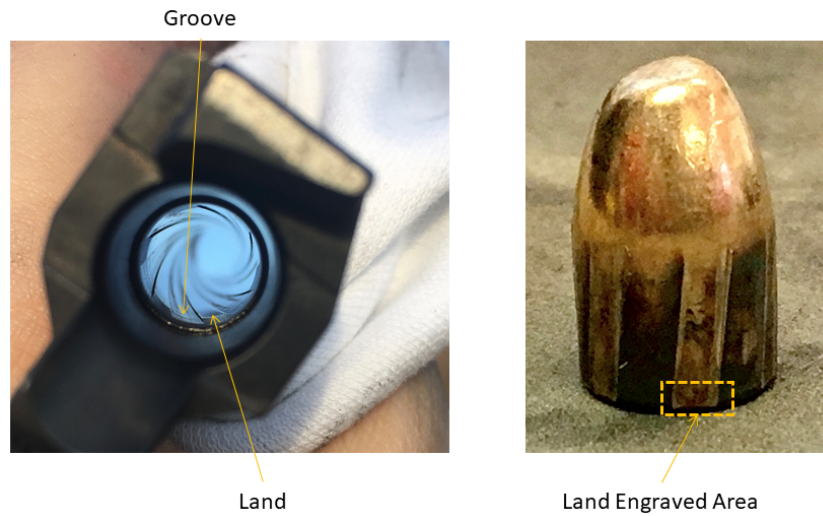


Figure 1: Photo of a traditionally rifled gun barrel (left) and a fired bullet (right).

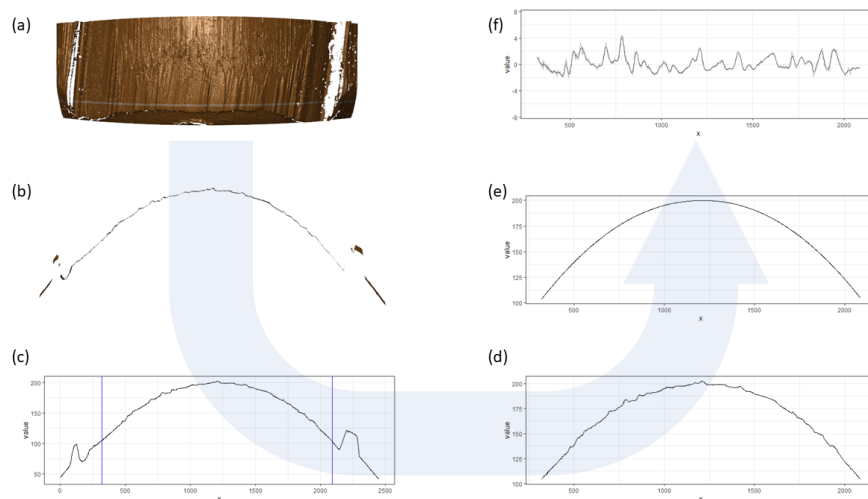


Figure 2: A framework of obtaining a bullet signature. (a) rendering from the 3D topographic scan of a land engraved area (LEA). The selected crosscut location is indicated by a thin white horizontal line. (b) view of the cross-section of the land engraved area at the white line in (a). (c) the crosscut data plotted in 2D; blue vertical lines indicate the position of the left and right grooves. (d) the crosscut data after chopping the left and right grooves. (e) the fitted curvature using LOESS. (f) after removing the curvature from the crosscut data, the bullet signature is obtained.

of these registration positions, a **congruent registration position** is identified, and the number of basis segments taking the congruent registration position is the CMPS score. Note that researchers in Chen et al. (2019) refer to the origin of basis segments as the reference, but in this paper we refer to it as the comparison signature. High CMPS scores are achieved between more similar signatures and are therefore indicative of a same-source pair. Low scores between pairs of signatures are attributed to different source pairs. However, a specific threshold of the CMPS score to distinguish between same-source and different-source comparisons is not provided in Chen et al. (2019), instead the threshold depends on the underlying structure of the data and the choice of parameters. In a legal setting this variability is problematic because it allows for situations in which experts could choose parameters based on whether they are witnesses for the defense or the prosecution. Further research is needed to understand how to determine optimal threshold settings. The CMPS algorithm can assist firearms examiners with drawing a conclusion about the source of a comparison pair. Thus, in this paper we present an open-source implementation of the CMPS algorithm in the R package **cmpsR** available from both CRAN and Github. This publicly available implementation calculates the CMPS score of a comparison using the following code:

```
# install.packages("cmpsR")

library(cmpsR)
data(bullets)

sig1 <- bullets$sigs[[2]]$sig
sig2 <- bullets$sigs[[9]]$sig
sig3 <- bullets$sigs[[10]]$sig

cmps_result_KM <- extract_feature_cmps(sig1, sig2)
cmps_result_KNM <- extract_feature_cmps(sig1, sig3)
```

In this example, the comparison between sig1 and sig2, two signatures coming from the same source (a known-match comparison), gets a CMPS score of 17; the comparison between sig1 and sig3, two signatures coming from different sources (a known non-match comparison), gets a CMPS score of 1.

We also implemented graphing tools for users to better understand these results as well as the algorithm itself.

The section “Implementation” will go through the algorithm and show how to use the **cmpsR** package. A further example that illustrates the main points is also included. In the section on “Evaluation metrics”, we introduce new CMPS metrics that summarize land-level CMPS scores and a sum of squares ratio that can be used to evaluate algorithmic results. The section “Results” presents the results of evaluating the **cmpsR** package using Hamby set 252 and Hamby set 44. The results from Hamby set 252 are used to verify that our implementation is, at least qualitatively, comparable to the algorithm described in Chen et al. (2019). Results from Hamby 44 show the need for a further investigation of the parameter choices even in the case of bullets fired from the same barrels. The last section covers some final discussion and conclusions.

Implementation

Algorithm

Conceptually, the CMPS algorithm consists of three main steps:

1. **cut the comparison signature into consecutive, non-overlapping, and equal-length basis segments:** The command `get_segs(x, len=50)` implements this step: it takes bullet signature `x` in the format of a numeric vector and cuts it into consecutive, non-overlapping and equal-length segments of length `len`, which are referred to as “basis segments”. Note that the parameter `len` determines the length of a basis segment and thus affects the total number of basis segments, which is the upper limit of the CMPS score of a comparison. The default value of `len` is 50, which will result in about 25 basis segments for the example data of the package.
2. **identify candidate positions:** For each basis segment a set of candidate registration positions on the comparison signature is identified based on the segment’s similarity to the reference signature. In the first step, the cross-correlation function of the segment to the reference is calculated, then a number of positions with high correlation values are identified as candidate positions. In case multiple segment lengths are considered, the length of each basis segment is expanded (by default it is doubled) and these two steps are repeated. Only when candidate

positions coincide (or are similar enough), they are considered further. Figure 5 and Figure 6 illustrate these ideas.

- **Calculate the cross-correlation curve:** Calculate the cross-correlation curve between a basis segment x and the reference signature y using the function `get_ccf4(x, y, ...)` as shown in Figure 5(b). The position indicates the lag by which a basis segment is moved with respect to its original placement. A position is considered “good” if it results in a peak in the cross-correlation between the basis segment and the reference.
 - **Correlation peaks:** Two strategies referred to as “multi-peak inspection” and “multi-peak inspection at different segment lengths” in Chen et al. (2019) are used for identifying positions of correlation peaks as candidate positions. The latter is also called the “multi-segment lengths strategy”. The parameter `npeaks_set` in `extract_feature_cmps(...)` determines which strategy to use and the number of candidate positions:
 - If `npeaks_set` is an integer vector of length 1, for example, `npeaks_set = 5`, the positions of the top five peaks in the cross-correlation curve are identified as candidate positions for registration.
 - If `npeaks_set` is an integer vector of length more than 1, for example, `npeaks_set = c(5, 3, 1)`, the multi-segment lengths strategy will be used: calculate the cross-correlation function between a basis segment and the reference and identify positions of the top five peaks; double the segment length to a specified value, re-calculate the cross-correlation function, and identify three peaks; repeat this process and identify a single peak in the newly computed cross-correlation function. Figure 6 shows an example of three levels of basis segment 6 and their corresponding cross-correlation curves and identified peaks. Note that in Chen et al. (2019) the segment length is doubled at each level of a basis segment, but in the present implementation users are allowed to choose the segment length at each level.
 - `get_ccr_peaks(comp, segments, seg_outlength, nseg = 1, npeaks = 5)` computes the cross-correlation curve between a basis or increased segment and the reference signature and finds peaks in the cross-correlation curve. The number of peaks detected is equal to `npeaks`, which is an integer. `segments`, `seg_outlength`, and `nseg` determine the segment in the cross-correlation computation, and `comp` gives the reference signature. If the multi-segment lengths strategy is used, then `get_ccr_peaks(...)` is called in a `lapply()` for each level of the basis segment. The resulting list is called `ccr_list`.
 - **multi-segment lengths strategy:** with the multi-segment lengths strategy being used, a position is identified as a candidate position for registration and is called a “consistent correlation peak” if it results in a top peak in the cross-correlation curve with a tolerance zone determined by T_x in all segment levels. Note that in Chen et al. (2019), a segment at its largest scale (highest level) always identifies one peak, but we do not have this requirement in our implementation.
 - the function `get_seg_scale(segments, nseg, out_length)` is used to obtain the (potentially increased) version of a basis segment. `segments`, which is a list containing all basis segments generated by the function `get_segs(...)` in step 1, and `nseg` are used to determine the basis segment to be increased. `out_length` specifies the length of the output segment.
 - `get_ccp(ccr_list, T_x = 25)` tries to identify the “consistent correlation peak”. `ccr_list` is the result of `lapply()` and `get_ccr_peaks(...)`, and T_x determines the size of a tolerance zone used in identifying the consistent correlation peak. `get_ccp(...)` returns NULL if there is no consistent correlation peak.
3. **determine the congruent registration position:** A candidate position “receives” votes from basis segments that identify it or a close position within a tolerance zone of T_x as a candidate position in step 2. Votes for all candidate positions are tallied, and the position with the highest number of votes gets chosen as the *congruent registration position*, indicating that most of the basis segments find their highly similar counterpart in the reference signature in terms of correlation at this registration position. In the case of ties, the middle position is taken as the congruent registration position. Basis segments with a congruent registration position are called “congruent matching profile segments” (CMPS). The total number of CMPS is the CMPS score of the comparison. `get_CMPS(input_ccp, T_x = 25)` is the function that tallies the votes and determines the congruent registration position and congruent matching profile segments (CMPS).

Note that there are several parameters in the CMPS algorithm that will affect the final results and are left to the users to decide, such as the length of basis segments `seg_length` in step 1, the number of

peaks `npeaks_set` identified on each level in step 2, and the length of the tolerance zone T_x in both step 2 and 3. In our implementation of the CMPS algorithm, we used the parameters given in the original CMPS paper (Chen et al. 2019) as the default values for these parameters. However, the authors state that no cross-validation has been done - there might also be issues with respect to the resolution of the scans. Further research is needed, until then users are advised to think of default values as starting values and consider alternatives. However, the evaluation framework based on the sum of squares ratio introduced in the later section could be used to evaluate the choices of these parameters. The main function that combines all steps in the CMPS algorithm described above is called `extract_feature_cmps(...)`. Here we present it with its default parameters.

```
extract_feature_cmps(
  x,
  y,
  seg_length = 50,
  Tx = 25,
  npeaks_set = c(5, 3, 1),
  include = NULL,
  outlength = NULL
)
```

The function `extract_feature_cmps` allows for the following input from users besides the previously discussed parameters `seg_length`, `npeaks_set`, and T_x :

- `x` and `y` are two signatures: `x` serves as the comparison signature (which will be divided into basis segments) and `y` is the reference signature;
- `include` determines the format of the function result. Besides the `CMPS_score`, other aspects of the comparison help in understanding how the `CMPS_score` is computed. By default `include` is set to `NULL` and only the `CMPS_score` is returned; further results are included when `include` is (an abbreviation of) one of or a vector of the following strings: `"nseg"`, `"congruent_pos"`, `"congruent_seg_idx"`, `"segments"`, `"parameters"`, and `"full_result"`. If `include` is specified as `"full_result"` (or its abbreviation), the output includes everything listed below.
 - `nseg`: the number of basis segments from the comparison signature; this is also the highest possible CMPS score of the comparison;
 - `congruent_pos`: the congruent registration position;
 - `congruent_seg_idx`: the indices of all congruent matching profile segments;
 - `ccp_list`: a list showing identified candidate positions of all basis segments;
 - `pos_df`: a data frame containing all candidate positions and their respective number of votes;
 - `segments`: a list containing all basis segments;
 - `parameters`: a list containing all input arguments of `extract_feature_cmps`;
- `outlength` specifies the segment length of a basis segment at each level under the multi-segment lengths strategy. By default `outlength` is set to `NULL`, indicating that a basis segment should double its segment length at the next level and conforming to the description in Chen et al. (2019).

In the remainder of the paper, we showcase the use of the **cmpsR** functionality on some examples and present the results of applying it to two datasets.

Installation

The **cmpsR** package is publicly available from CRAN and can be installed by

```
install.packages("cmpsR")
```

Moreover, its development version is also available from Github and can be installed by

```
# install.packages("remotes")
remotes::install_github("willju-wangqian/cmpsR")
```

An example

The **cmpsR** package contains a simple example to illustrate the basic usage of the package. The data in this example are twelve bullet signatures obtained from two bullets in Hamby set 252 (Hamby, Brundage, and Thorpe 2009). The procedure for generating signatures from high-resolution 3D

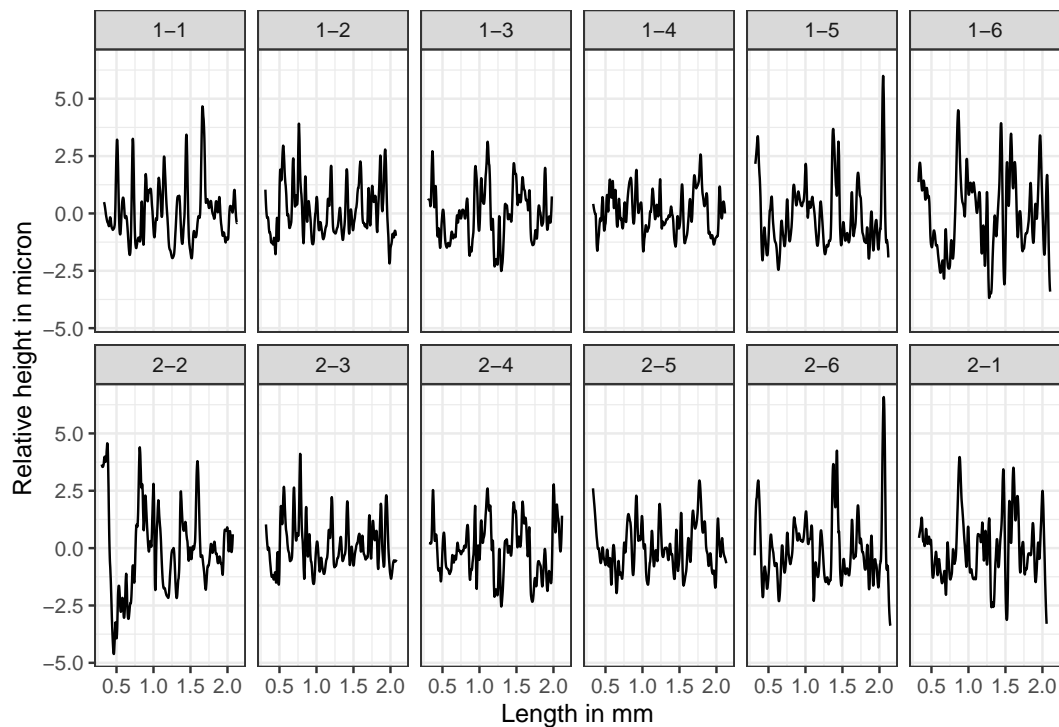


Figure 3: Signatures of all lands of bullet 1 in the top row, and of bullet 2 in the bottom row. Signatures in the second row are ordered to be in phase with the signatures above, i.e. matching signatures are displayed on top of each other. On the x-axis is the length of the scan in millimeter, and on the y-axis is the relative height in micron.

topographic scans of bullet lands used here follows the methodology described in Hare, Hofmann, and Carriquiry (2017) (as discussed above). The two bullets under consideration are known to have been fired from the same gun barrel, so for the 36 pairwise land-by-land comparisons, six comparisons are from same-source pairs (known matches) while thirty are from different-source pairs (known non-matches). To access the example data, we use

```
library(cmpsR)
data(bullets)
```

`bullets$sigs` is a list of twelve numeric vectors corresponding to the twelve bullet signatures shown in Figure 3. `bullets$source` contains the URLs to the corresponding x3p file containing the topographic scan from the NIST Ballistics Toolmark Research Database (Zheng 2016).

The signatures of Land 4 of Bullet 1 and Land 5 of Bullet 2 are stored in objects `sigs2` and `sigs1`, respectively. This comparison consists of a pair of signatures that are known to be a match – a KM (known match) comparison. We compute the CMPS score using two versions of the CMPS algorithm:

```
sigs1 <- bullets$sigs[bullets$bulletland == "2-5"][[1]]
sigs2 <- bullets$sigs[bullets$bulletland == "1-4"][[1]]

# compute cmps

# algorithm with multi-peak insepction at three different segment levels
cmps_with_multi_scale <-
  extract_feature_cmps(sigs1$sig, sigs2$sig,
    npeaks_set = c(5, 3, 1), include = "full_result"
  )

# algorithm with multi-peak inspection at the basis scale only
cmps_without_multi_scale <-
  extract_feature_cmps(sigs1$sig, sigs2$sig,
    npeaks_set = 5, include = "full_result"
  )
```


In the first example, `npeaks_set` is a vector of three integers, i.e. the algorithm uses the multi-segment lengths strategy to create the result object `cmps_with_multi_scale`. For `cmps_without_multi_scale` each basis segment is linked to the top 5 candidate positions. We use `include = "full_result"` to capture all results. In this example, the CMPS is 9 when using multiple segments, and 12 when using a single segment. As discussed in Chen et al. (2019), using multi-segment lengths strategy can reduce the number of false positives when identifying candidate positions; however, any score-based method is walking the line between false positives and false negatives. As the number of false positives is reduced the number of false negatives might rise. More discussion and comparisons between the two versions of the CMPS algorithm will be presented in later sections. Note that the multi-segment lengths method is slower because the algorithm is run once for each segment length.

Visualize and understand CMPS results

We also implemented graphing tools for visualizing the results of the CMPS algorithm. The goal is to provide users with tools to inspect each of the basis segments and to help them have a better understanding of how the algorithm works. Figure 4 shows the plots generated by the first graphing function, `cmps_signature_plot()`, and continues with the example above. `cmps_signature_plot()` takes the output of `extract_feature_cmps(..., include = "full_result")` and returns a list of 5 elements. It creates an overall impression of how the comparison signature aligns with the reference signature at the congruent registration position.

- The first element is a plot called `segment_shift_plot`, shown in Figure 4(a). On this plot the reference signature is drawn as a black line, congruent matching profile segments from the comparison signature are overlaid in red at the congruent registration position.

```
sig_plot <- cmps_signature_plot(
  cmps_with_multi_scale
)
```

```
# (a)
sig_plot$segment_shift_plot
```

- The second plot is called `signature_shift_plot`, shown in Figure 4(b). This visual presents both the comparison signature and the reference signature. The comparison signature is aligned with the reference signature based on the congruent registration position. Congruent matching profile segments are highlighted by solid red lines.

```
# (b)
sig_plot$signature_shift_plot
```

- Other elements of this list are `seg_shift` and `sig_shift`. `sig_shift` gives the congruent registration position, while `seg_shift` is a data frame showing the congruent matching profile segments and their identified candidate position closest to the congruent registration position.

```
sig_plot$seg_shift
#>   seg_idx seg_shift
#> 7         7         0
#> 8         8        -1
#> 14        14         5
#> 16        16         8
#> 17        17         8
#> 18        18         8
#> 19        19         9
#> 20        20         9
#> 22        22        12
```

While `cmps_signature_plot()` focuses on the signature level, `cmps_segment_plot()` focuses on the segment level. It provides the “full result” of `extract_feature_cmps()`, but also takes an argument, `seg_idx`, indicating which segment should be inspected. When checking `sig_plot$seg_shift` we notice that segment number 6 is not one of the congruent matching profile segments. We can therefore set `seg_idx = 6` in `cmps_segment_plot()` and investigate the reason why this segment disagrees with the congruent registration position.

For each segment scale, we have two plots: `segment_plot` and `scale_ccf_plot`, as shown in Figure 5 for the example of segment number 6:

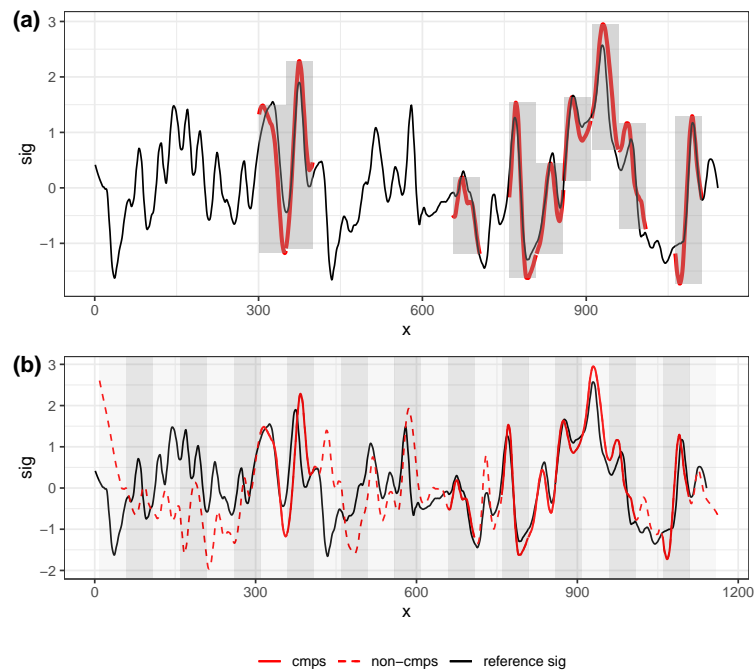


Figure 4: In (a) the black line shows the comparison signature; each red line segment shows one congruent matching profile segment. Each grey rectangle highlights one congruent matching profile segment. In (b) the black line shows the reference signature; the red line shows the comparison signature. Solid part shows the congruent matching profile segments, and the dashed part shows segments that do not agree with the congruent registration position.

- Figure 5(a) is the `segment_plot` for basis segment 6 at level one (in its original length). We used `npeaks_set = c(5, 3, 1)` in `extract_feature_cmps()` when calculating the CMPS score. Therefore the top five peaks are identified in the cross-correlation curve at level one. Segment 6 is plotted at the positions where these five peaks are identified with dashed lines in the `segment_plot`. The solid thick black line shows the segment at its original position (which in this example is very close to the actual registration position).

```
seg_plot <- cmps_segment_plot(
  cmps_with_multi_scale,
  seg_idx = 6
)
```

```
# (a)
seg_plot[[1]]$segment_plot
```

- Figure 5(b) is the `scale_ccf_plot` of basis segment 6 at level one. It shows the cross-correlation curve computed by the reference signature and the level-one basis segment 6. The five highest peaks are marked by dots on the curve.

```
# (b)
seg_plot[[1]]$scale_ccf_plot
```

Additionally, users can have more insights about why segment 6 is not a congruent matching profile segment if we put the `segment_plot` and `scale_ccf_plot` of all three segment levels together, as shown in Figure 6 with the help of `ggpubr::ggarrange()`.

```
library(ggpubr)

ggarrange(
  plotlist =
    unlist(seg_plot,
           recursive = FALSE
    ),
  ncol = 2,
  nrow = 3
)
```

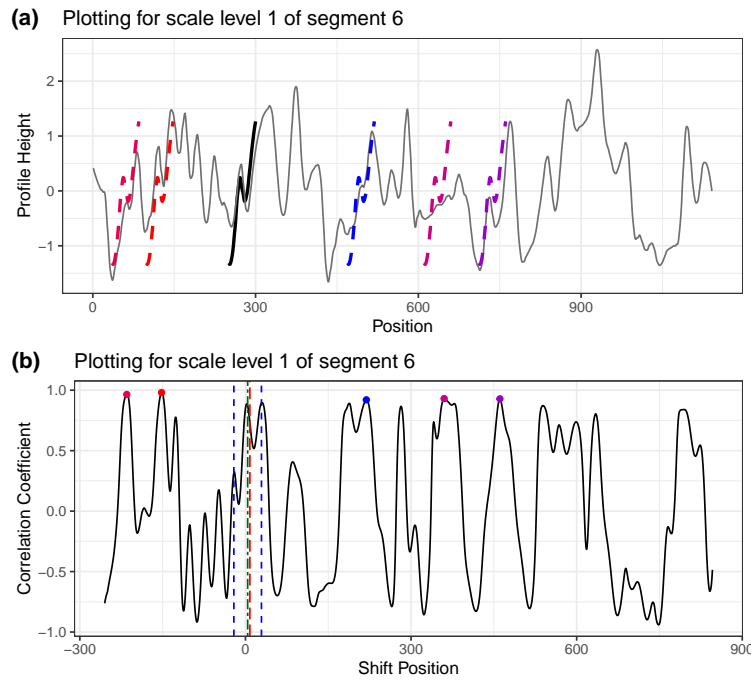


Figure 5: Plot (a) shows segment_plot for segment 6 at level one. The original position of segment 6 is indicated by the solid black line. Positions, where the segment achieves the 5 highest cross-correlations, are indicated by the dashed line segments. The scale_ccf_plot in plot (b) shows the cross-correlation curve between the reference signature and segment 6 at level one. The five highest peaks are marked by dots. The vertical red dashed line indicates the congruent registration position; the green dashed line shows a peak position in the highest segment level; the blue dashed lines show the tolerance zone around the green dashed line. We can see that none of the five highest peaks at level one falls within the tolerance zone, indicating that there is no consistent correlation peak or a candidate position identified by basis segment 6 under the multi-segment lengths strategy. Thus, the basis segment 6 doesn't vote for the congruent registration position and is not a cmps.

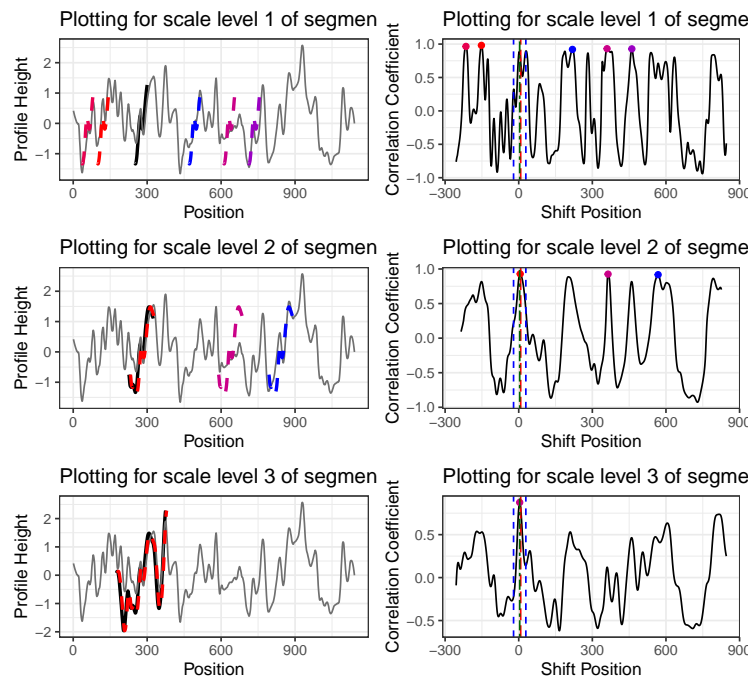


Figure 6: Put segment_plot and scale_ccf_plot of all three levels together. We are identifying the five highest peaks at level one, three peaks at level two, and one peak at level three since npeaks_set = c(5, 3, 1). The highest peak position at level three is marked by the green dashed line across all segment levels. However, the highest peak on level three does not coincide with any of the top five highest peaks at level one. This indicates that there is no consistent correlation peak or a candidate position for basis segment 6 under the multi-segment lengths strategy.

In Figure 6, the red vertical dashed line indicates the congruent registration position. We can see that the basis segment 6 does obtain a peak near the congruent registration position at level two and level three, respectively; however, this position doesn't give one of the five highest peaks at level one. As a result, segment 6 fails to identify the consistent correlation peak (ccp) and fails to become one of the congruent matching profile segments according to the multi-segment lengths strategy. The identified top five peaks at level one are also examples of "false positive" peaks. The "true positive" peak (the peak within the tolerance zone of the congruent registration position) is identified at level two and three by increasing the segment length, which justifies the usage of the multi-segment lengths strategy.

Evaluation metrics

Metrics based on CMPS scores

The CMPS algorithm measures the similarity between two signatures resulting in a similarity score of a land-to-level comparison. Bullets fired from traditionally rifled barrels have multiple land and groove engraved areas. Here, we are working with bullets fired from Ruger P85 barrels with six lands and grooves. A comparison of two bullets, therefore, involves 36 land-to-land comparisons, resulting in 36 CMPS scores (as shown in Figure 7). In order to obtain a single similarity score of a bullet-level comparison, we need to summarize these 36 CMPS scores. Two similarity metrics for bullet-level comparisons have been introduced in the literature (Chen et al. 2019): $CMPS_{max}$ and \overline{CMPS}_{max} . $CMPS_{max}$ is the highest CMPS score obtained among all land-level comparisons, while \overline{CMPS}_{max} is the highest possible mean CMPS score of land-level comparisons that are in the same phase:

In general, we assume each bullet has n land engravings (in our case $n = 6$). Let c_{ij} denote the CMPS score of a comparison between bullet 1 land i and bullet 2 land j , for $i, j = 1, \dots, n$. Let \mathcal{P}_k denote bullet land pairs in phase k for $k = 0, \dots, n - 1$, and

$$\mathcal{P}_k = \{(i, j) : i = 1, \dots, n; j = (i + k) \bmod n\} \tag{1}$$

, where mod denotes the modulo operation. For example, $\mathcal{P}_1 = \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1)\}$ when $n = 6$. Let k^* denote the index of the highest phase.

With that, the two measures to evaluate accuracy used in Chen et al. (2019) are defined as

$$CMPS_{max} = \max_{i,j} c_{ij}, \text{ and} \tag{2}$$

$$\overline{CMPS}_{max} = \frac{1}{n} \sum_{(i,j) \in \mathcal{P}_{k^*}} c_{ij}, \text{ where} \tag{3}$$

$$k^* = \arg \max_k \left[\frac{1}{n} \sum_{(i,j) \in \mathcal{P}_k} c_{ij} \right] \tag{4}$$

We can continue with the example used in previous sections. bullets contains bullet signatures of two bullets, bullet1 and bullet2. As mentioned before, each bullet has six land engravings, resulting in six bullet signatures. Thus, there are 36 pairwise bullet signature comparisons, resulting in 36 c_{ij} values in total. We use multi-segment lengths strategy with default parameters to compute these CMPS scores, and the result is shown in Figure 7. We can see that in this example,

$$CMPS_{max} = \max_{i,j} c_{ij} = 17$$

and since bullet lands in phase \mathcal{P}_1 gives the highest mean CMPS score ($k^* = 1$), we have

$$\begin{aligned} \overline{CMPS}_{max} &= \frac{1}{6} \sum_{(i,j) \in \mathcal{P}_1} c_{ij} \\ &= \frac{1}{6} (c_{12} + c_{23} + c_{34} + c_{45} + c_{56} + c_{61}) \\ &= \frac{1}{6} (3 + 17 + 14 + 10 + 15 + 16) \\ &= 12.5 \end{aligned}$$

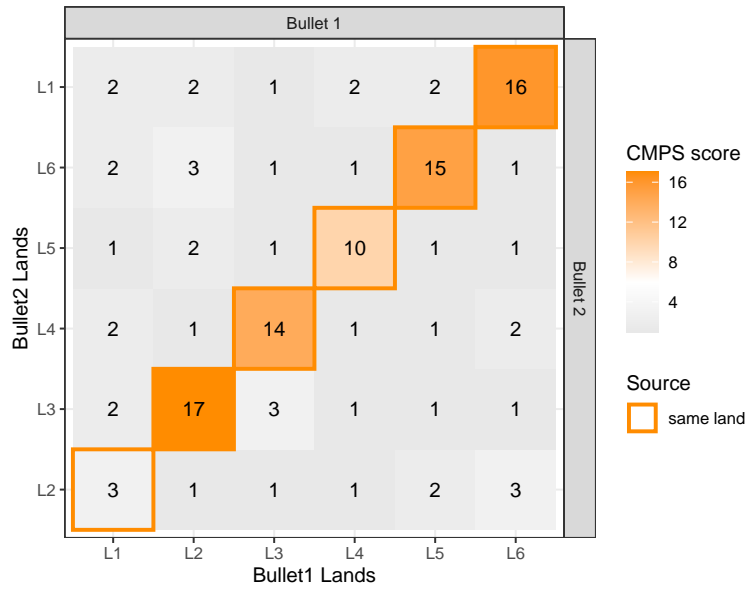


Figure 7: CMPS scores of all 36 pairwise bullet signature comparisons for two bullets. Land engraving pairs generated by the same land (KM comparisons) are highlighted. Note that in this example the axis along Bullet 2 starts with Land 2. This corresponds to Phase 1 in equation (1).

However, both $\overline{\text{CMPS}}_{\max}$ and $\overline{\text{CMPS}}_{\max}$ consider only relatively high CMPS scores and ignore the rest. So we introduce a new metric based on CMPS scores called $\overline{\text{CMPS}}_{\text{diff}}$, which is the difference between $\overline{\text{CMPS}}_{\max}$ and the mean of all other CMPS scores. With our notation above, we have:

$$\overline{\text{CMPS}}_{\text{diff}} = \left[\frac{1}{n} \sum_{(i,j) \in \mathcal{P}_{k^*}} c_{ij} \right] - \left[\frac{1}{n(n-1)} \sum_{(i,j) \notin \mathcal{P}_{k^*}} c_{ij} \right] \tag{5}$$

$\overline{\text{CMPS}}_{\text{diff}}$ highlights the difference between CMPS scores of matching and non-matching comparisons. If two bullets are non-matching, all 36 CMPS scores are expected to be small with relatively the same values, resulting in a $\overline{\text{CMPS}}_{\text{diff}}$ value close to 0. For the example above, $\overline{\text{CMPS}}_{\text{diff}} = 12.5 - 1.53 = 10.97$

Scaled CMPS scores

Another issue with the CMPS score is that the highest possible CMPS score (the total number of basis segments) might differ across comparisons (as shown in Figure 8(a)) due to different lengths of bullet signatures and different lengths of basis segments specified by the parameter. A CMPS score of 5 might indicate a non-match if the highest possible CMPS score is 30 but indicate a match if the highest possible CMPS score is 6. Thus, we introduce the scaled CMPS score, denoted as c_{ij}^* . Let s_{ij} denote the highest possible CMPS score or the total number of basis segments, then the scaled CMPS score c_{ij}^* is defined as the ratio of raw score and maximum score:

$$c_{ij}^* = \frac{c_{ij}}{s_{ij}} \tag{6}$$

The scaled CMPS scores of the above example are shown in Figure 8(b). Compared to the original CMPS scores, scaled scores have values within the interval $[0, 1]$ regardless of the length of the basis segments and therefore make a comparison of values possible across different parameter choices. Similar to the original CMPS scores, we will denote the scaled CMPS scores adjusted for out-of-phase background values by $\overline{\text{CMPS}}_{\text{diff}}^*$. For example, $\overline{\text{CMPS}}_{\text{diff}}^* = 0.498$ for Figure 8(b)

Sum of squares ratio

The “sum of squares ratio” quantifies how well two groups of values separate. Let n_T denote the total number of observations, n_k denote the number of observations in group k , and y_{kl} denote the l -th observation in group k , for $k = 1, 2$ and $l = 1, \dots, n_k$. Let $\bar{y}_k = \frac{1}{n_k} \sum_{l=1}^{n_k} y_{kl}$ denote the mean value in group k and $\bar{y}_{..} = \frac{1}{n_T} (\sum_k \sum_{l=1}^{n_k} y_{kl})$ denote the mean value of all observations. Consider the following model:

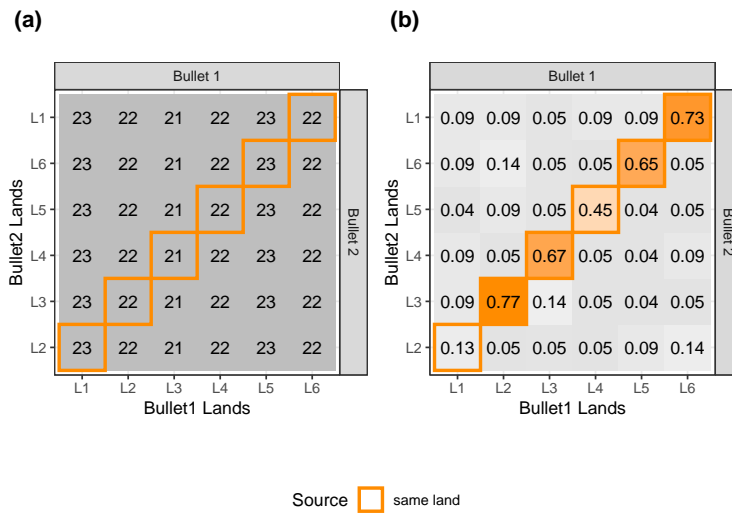


Figure 8: Plot (a) shows the highest possible CMPS scores (the total number of basis segments) for the 36 comparisons. (b) shows the scaled CMPS scores for the 36 comparisons.

$$y_{kl} = \mu_k + e_{kl} \tag{7}$$

where μ_k is the true mean of group k and e_{kl} is a random effect of different observations. Then we can define the sum of squares ratio V as:

$$V = \frac{\sum_k n_k (\bar{y}_k - \bar{y}_{..})^2}{\sum_k \sum_l^{n_k} (y_{kl} - \bar{y}_k)^2} \tag{8}$$

The numerator of the sum of squares ratio V quantifies the variation between the two groups, while the denominator quantifies the variation within each group. The sum of squares ratio V can be used as an index for evaluating scans, metrics, and different sets of parameters if the same data set is being used. Some examples will be presented in the following section. If we impose the normality and independence assumptions on the random effects e_{kl} , the sum of squares ratio V becomes a scaled F-statistic with degrees of freedom of $k - 1$ and $n_T - k$ and $F = \frac{n_T - k}{k - 1} V$. If we want to compare different data sets, stating that a certain setup can achieve better separation on one data set than another, we can scale the sum of squares ratio V and obtain the F-statistic and obtain the corresponding p-value as an index for comparison.

Using the sum of squares ratio V as an evaluation metric, we are able to construct a pipeline that aims to find the optimal parameter values for the CMPS algorithm by maximizing the sum of squares ratio. Note that other measures of an algorithm, such as the accuracy and the AUC (Area Under the Curve), are also important and useful. But when algorithms achieve 100 percent accuracy and AUC value of 1, we need other measures such as the sum of squares ratio to further distinguish the performance of algorithms. In the following section, we will use the sum of squares ratio to compare the CMPS metrics introduced earlier and investigate the effects of different parameter settings.

Results

As presented in the work of Chen et al. (2019), researchers applied the CMPS method to scans of one of the Hamby sets. While it is not explicitly stated in the paper, we presume this to be Hamby 252, as only those scans were publicly available at the time. In order to show that our implementation of the CMPS algorithm is able to reproduce the results in Chen et al. (2019) and be used for other data sets, we applied our implementation to both Hamby set 252 and Hamby set 44. Here we present how we obtained bullet signatures from the Hamby set data: for both Hamby 252 and Hamby 44, we started with scans in the form of x3p files in the database. Following the framework proposed by Hare, Hofmann, and Carriquiry (2017), we used the same set of parameters, removed damaged bullet scans, obtained bullet signatures for each bullet land engraving, and removed outliers in bullet signatures. Note that researchers of Chen et al. (2019) applied the CMPS algorithm to bullet signatures as well but used a framework different from ours. However, since their work is not open-source, we were not able

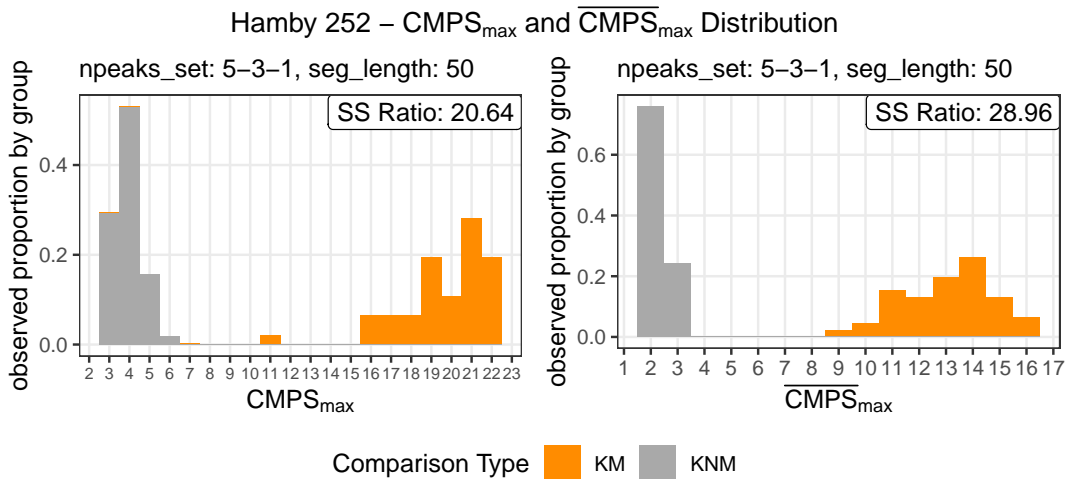


Figure 9: Distribution of $CMPS_{max}$ and \overline{CMPS}_{max} for Hamby 252; outliers are removed in bullet signatures; $seg_length = 50$, $T_x = 25$, $npeaks_set = c(5,3,1)$; instead of showing the counts on the y-axis, we present the observed proportions conditioned on KM group and KNM group to enhance the visibility of the bars.

to follow their framework and were only able to reproduce the results for Hamby set 252 qualitatively.

Hamby 252

Figure 9 shows the distribution of $CMPS_{max}$ and \overline{CMPS}_{max} after we applied the CMPS algorithm to Hamby set 252 with the multi-segment lengths strategy. The parameters we used in `extract_feature_cmps` for the CMPS algorithm are:

```
extract_feature_cmps(
  x, y,
  seg_length = 50,
  Tx = 25,
  npeaks_set = c(5, 3, 1),
  include = "nseg"
)
```

As noted above, the CMPS scores we found here are not exactly the same as those presented in Chen et al. (2019) since we were not able to follow their framework, but the results presented in Figure 9 are qualitatively equivalent to those presented in Chen et al. (2019), showing a clear separation between scores based on comparisons from known matches (KM) and scores from comparisons of known non-matches (KNM) for both $CMPS_{max}$ and \overline{CMPS}_{max} .

Additionally, to mimic the parameters used in Chen et al. (2019), we set $seg_length = 50$ and $T_x = 25$ to make sure that each basis segment has a length of $78.125 \mu m$ and the tolerance zone is $\pm 39.0625 \mu m$ (one unit represents $1.5625 \mu m$ for Hamby set 252).

The sum of squares ratios are 20.64 and 28.96 for $CMPS_{max}$ and \overline{CMPS}_{max} , respectively. This indicates that even though scores from $CMPS_{max}$ for known-match comparisons are larger than scores from the averaged version of \overline{CMPS}_{max} , these scores achieve a better separation between the two groups of comparisons.

Hamby 44

Similar procedures are applied to Hamby set 44, and Figure 10 shows the distribution of $CMPS_{max}$ and \overline{CMPS}_{max} , respectively. The parameters used in `extract_feature_cmps` are:

```
extract_feature_cmps(
  x, y,
  seg_length = 61,
  Tx = 30,
  npeaks_set = c(5, 3, 1),
  include = "nseg"
)
```

Since the resolution of Hamby 44 is set to $1.29 \mu m$ per unit, we make $seg_length = 61$ and $T_x = 30$ to ensure that the setup of Hamby set 44 is similar to that of Hamby set 252, resulting in basis segments of $78.69 \mu m$ and the tolerance zone of $\pm 38.7 \mu m$.

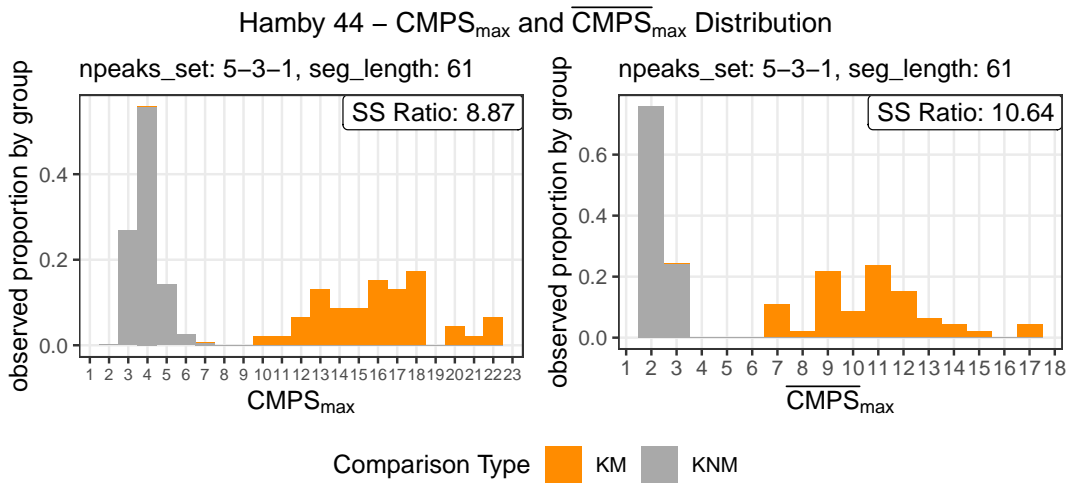


Figure 10: Distribution of $CMPS_{max}$ and \overline{CMPS}_{max} for Hamby 44; outliers are removed in bullet signatures; $seg_length = 61$, $T_x = 30$, $npeaks_set = c(5,3,1)$; instead of showing the counts on the y-axis, we present the observed proportions conditioned on KM group and KNM group to enhance the visibility of the bars.

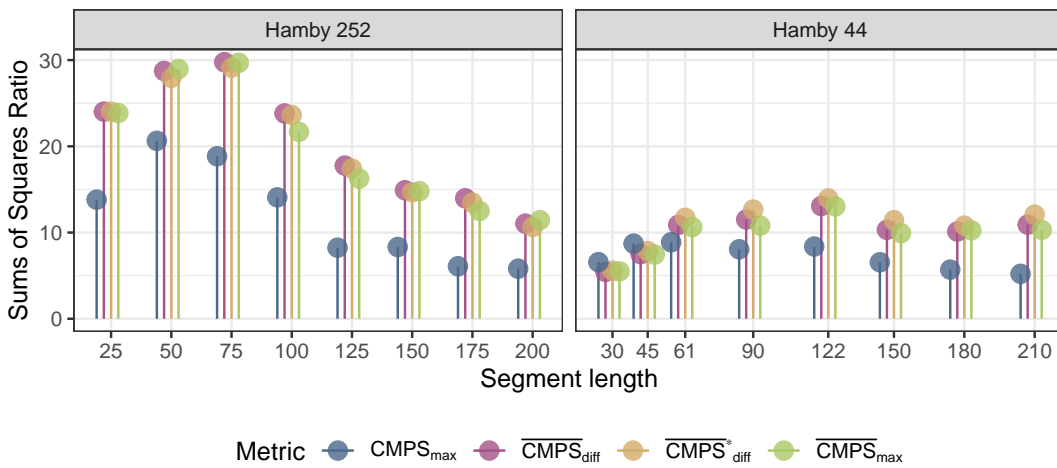


Figure 11: Comparison of results from the CMPS algorithm based on different basis segment lengths (parameter seg_length). Only the $CMPS_{max}$ metric suggests that the default values for the basis segment length result in the best separation. Better separation is achieved based on the modified CMPS metrics, including the newly suggested ones. For Hamby 252 these metrics agree on a segment length of 75, and a segment length of 122 for Hamby 44 yields better results.

As shown in Figure 10, again, we are able to see a clear separation between the known match comparisons and the known non-match comparisons, even though the separation is relatively small compared with that of Hamby set 252, which is also indicated by the sum of squares ratios. For this specific set of parameters, the sum of squares ratios are 8.87 and 10.64 for $CMPS_{max}$ and \overline{CMPS}_{max} , respectively. This might suggest that we could enlarge the separation in terms of the sum of squares ratio by using other CMPS metrics and other sets of parameters.

Comparing CMPS metrics and parameters

We investigated the effects of different sets of parameters in the example of both Hamby sets 252 and 44. More specifically, we investigated the separation achieved using the **cmpr** implementation under various segment lengths (controlled by the parameter seg_length). Specifically, we fixed the parameter $npeaks_set$ that controls the number of peaks at each segment level to be $npeaks_set = c(5, 3, 1)$ and modified the parameter seg_length . Figure 11 shows that the default values of seg_length for Hamby 252 and Hamby 44 (50 for Hamby 252 and 61 for Hamby 44, which represent $78.125 \mu m$ and $78.69 \mu m$, respectively) result in high values of the sum of squares ratio, no matter which CMPS metrics are used for an evaluation. However, we also see, that for Hamby set 252 a basis segment length of 75 is a better choice than the default segment length; a basis segment length of 122 results in a higher value of the sum of squares ratio for Hamby 44.

Figure 12 shows the results of the CMPS algorithm using different strategies for identifying peaks

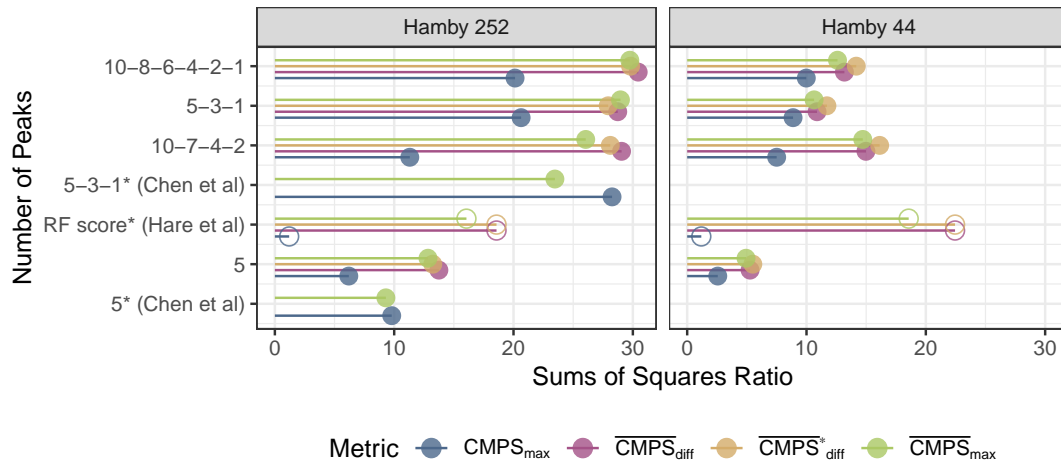


Figure 12: Comparison of CMPS results based on different strategies of number of peak selections. Starred results compare CMPS performance with results published in the literature. Results for the random forest score are represented with circles because the metrics are computed not based on the CMPS scores, but on the random forest scores with the same logic. Since random forest scores lie within the interval $[0, 1]$, scaling the random forest scores will not change the results.

in the correlation structure between signatures. Basis segment length is fixed to the default level for this evaluation. As can be seen in Figure 12, the default value of `npeaks_set` (`npeaks_set = c(5, 3, 1)`) leads to promising results in terms of the sum of squares ratio; however, other choices of `npeaks_set` match and exceed this sum of squares ratio value.

As seen before, the results suggest that the $\overline{\text{CMPS}}_{\text{max}}$ metric produces the least amount of separation compared with the other three CMPS metrics. They also suggest that the two newly proposed metrics, $\overline{\text{CMPS}}_{\text{diff}}$ and $\overline{\text{CMPS}}^*_{\text{diff}}$, lead to equally good or even better results as $\overline{\text{CMPS}}_{\text{max}}$. Because $\overline{\text{CMPS}}^*_{\text{diff}}$ relies on a scaled version of CMPS scores, it is more comparable to other similarity scores and summarizes the out-of-phase background CMPS scores, making it superior to the other CMPS metrics.

What we can also observe in Figure 11 and Figure 12 is that the values of the sum of squares ratio for Hamby 44 is lower than those for Hamby 252. This might be because determining the source is a harder task for Hamby 44 than for Hamby 252, but also suggests that the choice of parameters also depends on the resolution or the scanning process of the data set. The same set of parameters might work for one data set, but not work equally well for another.

The purpose of the results shown in Figure 11 and Figure 12 is not to determine the “best” parameters for the CMPS algorithm, but to show that the sum of squares ratio can be used as an evaluation measure to compare different parameters, metrics, or scans. A pipeline that maximizes the sum of squares ratio might help researchers determine the set of parameters that work best for their data. But a large database that is representative is what we really need in order to fully understand and cross-validate the parameters of the CMPS algorithm.

Comparing with original results and the random forest model

Chen et al. (2019) present histograms of $\overline{\text{CMPS}}_{\text{max}}$ and $\overline{\text{CMPS}}^*_{\text{max}}$ for `npeaks_set = 5` and `npeaks_set = c(5, 3, 1)` with (presumably) Hamby 252. The values in these histograms allow us to calculate the sum of squares ratios and include the results in Figure 12 as well. They are marked by an asterisk at the top right corner in Figure 12. The sum of squares ratios we obtained for `npeaks_set = 5` and `npeaks_set = c(5, 3, 1)` is slightly higher than those obtained from the histograms of Chen et al. (2019). It’s curious to see that for the Hamby 252 results published in Chen et al. (2019) the $\overline{\text{CMPS}}_{\text{max}}$ metric achieves values of the sum of squares ratio higher than those achieved by the $\overline{\text{CMPS}}^*_{\text{max}}$ metric.

Since the researchers of Chen et al. (2019) did not use $\overline{\text{CMPS}}_{\text{diff}}$ or $\overline{\text{CMPS}}^*_{\text{diff}}$, and they did not apply the CMPS algorithm to the Hamby set 44, we were not able to compare those results.

In Figure 12, we also included the sum of squares ratios computed from the random forest scores (Hare, Hofmann, and Carriquiry 2017) for different metrics. The random forest model presented in Hare, Hofmann, and Carriquiry (2017) was trained at the Center for Statistics and Applications in Forensic Evidence (CSAFE) and is publicly available in the R package `bulletxtctr` (Hofmann, Vanderplas, and Krishnan 2019). Similar to the CMPS algorithm, this random forest model produces a score to quantify the similarity of a land-by-land comparison. The RF scores lie in an interval of $[0,1]$ making them most comparable to the scaled CMPS scores. We applied the logic of $\overline{\text{CMPS}}_{\text{max}}$, $\overline{\text{CMPS}}^*_{\text{max}}$, and $\overline{\text{CMPS}}_{\text{diff}}$ to the random forest scores and obtained results of the random forest model

for different metrics. As shown in Figure 12, the random forest model does not achieve the sum of squares ratio as high as those achieved by the top CMPS algorithms for Hamby 252, but it does achieve better results for Hamby 44. This might suggest that inclusion of the CMPS score as an additional feature in the random forest model training might be beneficial for an overall separation to determine the source.

For details about the implementation, examples, and results, please refer to the “Supplementary materials”.

Conclusion

In this paper, we present the **cmpsR** package, an open-source implementation of the Congruent Matching Profile Segments (CMPS) algorithm (Chen et al. 2019), and apply it to two datasets in Hamby study (Hamby, Brundage, and Thorpe 2009) to show its potential for further research. The CMPS algorithm was proposed by NIST in 2019 and was made for objective tool marks comparisons. We introduce the basic logic of the CMPS algorithm and layout its implementation in the **cmpsR** package. We also showcase the functionality of the **cmpsR** package with a small dataset example that is included in the package. In the **cmpsR** package we implement some graphing tools for users to visualize results and to gain a better understanding of both the algorithm and the results.

Additionally, we propose two new metrics based on the CMPS scores and compare the new metrics with the existing metrics. We also introduce a principled evaluation framework of algorithmic results using a measure based on the sum of squares ratio. We showcase the implementation with two datasets in the Hamby study (Hamby set 252 and Hamby set 44) and compare the CMPS algorithm using different sets of parameters and different metrics, following the evaluation framework based on the sum of squares ratio. The results obtained are promising: we were both able to reproduce the results in Chen et al. (2019) qualitatively and achieve a clear separation between the known match (KM) comparisons and known non-match (KNM) comparisons in another bullet study. However, the comparisons among different sets of parameters suggest that the optimal choice for parameter settings varies between different datasets.

Note, that the main difference between Hamby sets 44 and 252 is that they were taken with different resolution scanning devices. The bullets for Hamby 44 and Hamby 252 are fired from the same ten consecutively rifled P-85 Ruger barrels. The difference in parameter settings for optimal differentiation between same-source comparisons and different-source comparisons is therefore quite surprising. In the next steps, we need validation studies similar to Vanderplas et al. (2020) - trying out the CMPS algorithm on different firearms and ammunition combinations to test the limits of the algorithm. The evaluation framework we proposed based on the sum of squares ratio will facilitate such studies and other validation of the algorithm.

Comparisons of the R implementation of the CMPS algorithm with the random forest model proposed by Hare, Hofmann, and Carriquiry (2017) suggest that adding the CMPS score as an additional feature in the random forest model might add further separation between known match (KM) comparisons and known non-match (KNM) comparisons.

The open-source implementation of the CMPS algorithm provided by the **cmpsR** package is just one step towards to the framework of open science. Open science is particularly important to fields like forensic science where transparency and accuracy are critical to fair justice. Open-source implementations not only allow a peer-review but also facilitate further research, such as parameter cross-validations, method comparisons, and the development of statistical methods for modeling KM and KNM CMPS score distributions, which can then be used for error-rate estimations and fair applications that were called for by the PCAST (President’s Council of Advisors on Science and Technology 2016) report. However, having open-source implementations is not enough for open science. In order to actually build the world of open science, many other efforts, such as open evaluation, open access publication, open educational resources, etc., are still needed. The database built and maintained by the National Institute of Standards and Technology is a good example of open data. What we are aiming for is an open system that is able to collect open results, compare multiple algorithms using multiple datasets, and evaluate algorithmic variation and accuracy. The evaluation metric we proposed in this paper can be used to compare different algorithms or even different datasets and is a snippet of this open system. But the core of this open system is the open science culture and contributions of the community.

Acknowledgement

This work was funded (or partially funded) by the Center for Statistics and Applications in Forensic Evidence (CSAFE) through Cooperative Agreements 70NANB15H176 and 70NANB20H019 between

NIST and Iowa State University, which includes activities carried out at Carnegie Mellon University, Duke University, University of California Irvine, University of Virginia, West Virginia University, University of Pennsylvania, Swarthmore College and University of Nebraska, Lincoln.

Supplementary materials

The zip file “supplementary-files.zip” contains R scripts for reproducing all aspects of this paper.

Data not included in “supplementary-files.zip”

The processed data of Hamby set 252 and Hamby set 44 are stored in two .rds files:

- [BulletSignatures252.rds](#)
- [BulletSignatures44.rds](#)

And the ground truth of Hamby set 252 is provided in [StudyInfo.xlsx](#)

Due to the size of the file, these processed data are not included in the “supplementary-files.zip”, but the links are provided for download.

In order to reproduce the results in the paper, please download “BulletSignatures252.rds”, “BulletSignatures44.rds”, and “StudyInfo.xlsx” and save them in a folder named “bullet_signatures_etc”. And place the folder “bullet_signatures_etc” into the folder of all the R scripts of “supplementary-files.zip”.

Please check out the folder structure of [the reproducible folder](#) for reference.

File description

- `hamby*_result_generator.R`: these R scripts take the processed data of Hamby set 252 and Hamby set 44, generate preliminary results used in the paper, and save the results in the .csv format in the folder data-csv. Please make sure that the package versions of `bulletxrctr` and `cmpsR` meet the requirements.
- `rds_generator.R`: this R script takes the generated .csv files and produce `CMPSpaper_results.rds`. `CMPSpaper_results.rds` is identical to `data/CMPSpaper_results.rds` and is used generate figures and other results presented in the paper.

Data included in “supplementary-files.zip”

- `data-csv`: this folder contains .csv files we generated using R scripts `hamby*_result_generator.R`
- `CMPSpaper_results.rds`: this is the .rds file we generated using R script `rds_generator.R`

These data can be used as reference or example results of the reproducible codes

- `csafe_rf2.rds`: this .rds file contains the random forest model used in this paper

References

- AFTE Criteria for Identification Committee. 1992. “Theory of Identification, Range Striae Comparison Reports and Modified Glossary Definitions.” *AFTE Journal* 24 (3): 336–40.
- Brundage, David J. 1998. “The Identification of Consecutively Rifled Gun Barrels.” *AFTE Journal* 30 (3): 438–44.
- Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2021. *shiny: Web Application Framework for R*. <https://CRAN.R-project.org/package=shiny>.
- Chen, Zhe, Wei Chu, Johannes A. Soons, Robert M. Thompson, John Song, and Xuezheng Zhao. 2019. “Fired bullet signature correlation using the Congruent Matching Profile Segments (CMPS) method.” *Forensic Science International* 305: Article 109964, (10 pages). <https://doi.org/https://doi.org/10.1016/j.forsciint.2019.109964>.
- Chumbley, L. Scott, Max D Morris, M. James Kreiser, Charles Fisher, Jeremy Craft, Lawrence J Genalo, Stephen Davis, David Faden, and Julie Kidd. 2010. “Validation of Tool Mark Comparisons Obtained Using a Quantitative, Comparative, Statistical Algorithm: VALIDATION OF TOOL MARK COMPARISONS.” *Journal of Forensic Sciences* 55 (4): 953–61. <https://doi.org/https://doi.org/10.1111/j.1556-4029.2010.01424.x>.
- Cleveland, William S., Eric Grosse, and William M. Shyu. 1991. “Local Regression Models.” In *Statistical Models in s*, edited by John M. Chambers and Tibshirani J. Hastie, 309–76. Boca Raton, Florida: Chapman; Hall/CRC.

- Committee on Identifying the Needs of the Forensic Sciences of the National Research Council. 2009. "Strengthening Forensic Science in the United States: A Path Forward." <https://www.ncjrs.gov/pdffiles1/nij/grants/228091.pdf>; National Academies Press.
- Hamby, James E., David J. Brundage, Nicholas D. K. Petraco, and James W. Thorpe. 2019. "A Worldwide Study of Bullets Fired From 10 Consecutively Rifled 9mm RUGER Pistol Barrels—Analysis of Examiner Error Rate." *Journal of Forensic Sciences* 64 (2): 551–57. <https://doi.org/10.1111/1556-4029.13916>.
- Hamby, James E., David J. Brundage, and James W. Thorpe. 2009. "The Identification of Bullets Fired from 10 Consecutively Rifled 9mm Ruger Pistol Barrels: A Research Project Involving 507 Participants from 20 Countries." *AFTE Journal* 41 (2): 99–110.
- Hare, Eric, Heike Hofmann, and Alicia Carriquiry. 2017. "Automatic Matching of Bullet Land Impressions." *Ann. Appl. Stat.* 11 (4): 2332–56. <https://doi.org/10.1214/17-AOAS1080>.
- Hofmann, Heike, Susan Vanderplas, and Ganesh Krishnan. 2019. *Bulletxtctr: Automatic Matching of Bullet Striae*. <https://heike.github.io/bulletxtctr/>.
- Hofmann, Heike, Susan Vanderplas, Ganesh Krishnan, and Eric Hare. 2020. *x3ptools: Tools for Working with 3D Surface Measurements*. <https://github.com/heike/x3ptools>.
- Krishnan, G., and H. Hofmann. 2019. "Adapting the Chumbley Score to Match Striae on Land Engraved Areas (LEAs) of Bullets." *J Forensic Sci* 64 (3): 728–40.
- President's Council of Advisors on Science and Technology. 2016. "Report on Forensic Science in Criminal Courts: Ensuring Scientific Validity of Feature-Comparison Methods." https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/PCAST/pcast_forensic_science_report_final.pdf.
- Song, J., L. Ma, E. Whinton, and T. Vorburger. 2005. "2d and 3d Surface Texture Comparisons Using Autocorrelation Functions." In *Measurement Technology and Intelligent Instruments VI*, 295:437–40. Key Engineering Materials. Trans Tech Publications Ltd. <https://doi.org/10.4028/www.scientific.net/KEM.295-296.437>.
- United States Department of Justice, Federal Bureau of Investigation. n.d.a. "Crime in the United States, 2019." <https://ucr.fbi.gov/crime-in-the-u.s/2019/crime-in-the-u.s.-2019/tables/expanded-homicide-data-table-8.xls>.
- . n.d.b. "Expanded Homicide Tables, 2020." <https://s3-us-gov-west-1.amazonaws.com/cg-d4b776d0-d898-4153-90c8-8336f86bdfec/CIUS/downloads/2020/expanded-homicide-2020.zip>.
- Vanderplas, Susan, Melissa Nally, Tylor Klep, Cristina Cadevall, and Heike Hofmann. 2020. "Comparison of Three Similarity Scores for Bullet LEA Matching." *Forensic Science International* 308: 110167. <https://doi.org/https://doi.org/10.1016/j.forsciint.2020.110167>.
- Zheng, Xiaoyu Alan. 2016. "NIST Ballistics Toolmark Research Database (NBTRB)." <https://tsapps.nist.gov/NRBD>.

Wangqian Ju
Department of Statistics
Center for Statistics and Applications in Forensic Evidence
Iowa State University
2438 Osborn Dr
Ames, IA 50011
<https://github.com/willju-wangqian>
ORCID: 0000-0002-9977-377X
wju@iastate.edu

Heike Hofmann
Department of Statistics
Center for Statistics and Applications in Forensic Evidence
Iowa State University
2438 Osborn Dr
Ames, IA 50011
<https://github.com/heike>
ORCID: 0000-0002-9079-593X
hofmann@iastate.edu

rassta: Raster-Based Spatial Stratification Algorithms

by Bryan A. Fuentes, Minerva J. Dorantes, and John R. Tipton

Abstract Spatial stratification of landscapes allows for the development of efficient sampling surveys, the inclusion of domain knowledge in data-driven modeling frameworks, and the production of information relating the spatial variability of response phenomena to that of landscape processes. This work presents the **rassta** package as a collection of algorithms dedicated to the spatial stratification of landscapes, the calculation of landscape correspondence metrics across geographic space, and the application of these metrics for spatial sampling and modeling of environmental phenomena. The theoretical background of **rassta** is presented through references to several studies which have benefited from landscape stratification routines. The functionality of **rassta** is presented through code examples which are complemented with the geographic visualization of their outputs.

1 Introduction

The application of robust, quantitative approaches for the spatial modeling of environmental phenomena has increased in the past few decades mainly due to an increase in computational power, advances in statistical modeling, and the availability of geospatial layers of environmental information (Scull et al. 2003; Elith and Leathwick 2009). Most of these approaches aim at building explicit quantitative relationships between environmental controls and response phenomena through statistical learning. Examples of these approaches include digital soil mapping (DSM) (McBratney, Mendonça Santos, and Minasny 2003), species distribution modeling (SDM) (Guisan and Zimmermann 2000), land use/land cover classification (Ham et al. 2005), and forest fire modeling (Chuvieco et al. 2010). Despite the extensively documented success of these approaches, there are still some challenges that limit their application. For instance, poor statistical performance is often reported in studies where input data is too limited to accurately represent control-response relationships (Araújo and Guisan 2006). Moreover, model parsimony and interpretation of results can be compromised when using ‘black-box’ algorithms (Arrouays et al. 2020). Similarly, including *a priori* knowledge about natural processes in purely statistical approaches can be challenging to achieve (Heuvelink and Webster 2001).

Several studies have suggested embedding spatial stratification routines within approaches such as DSM, SDM, land use/cover mapping, forest fire modeling, and others to overcome the challenges limiting their application. In such studies, the spatial stratification of landscapes creates units with reduced spatial variability of environmental phenomena as compared to the overall variability across a landscape. The use of these units allows the researcher to (a) obtain balanced representations of control-response relationships (Guisan and Zimmermann 2000; West et al. 2016); (b) include expert knowledge of physical processes for improving modeling with limited data (Zhu et al. 2008); (c) improve the performance of parameterization of mechanistic models (Park and Van De Giesen 2004; Baldwin, Naithani, and Lin 2017); and, (d) facilitate the interpretation of environmental conditions and their influence on the spatiotemporal variability of processes of interest (Rodrigues et al. 2019).

In general, landscape stratification routines follow fundamental ecological concepts that explain the hierarchical and multi-scale nature of relationships between environmental phenomena across space (Allen and Starr 1982). Therefore, landscape stratification methods have been applied in many studies that use geospatial information for environmental modeling, such as those previously cited. However, few packages exist in the R environment with functions strictly aimed at landscape stratification routines using geospatial data. Although one could implement custom stratification algorithms using multiple all-purpose geospatial analysis packages such as **terra** (Hijmans 2021) and **sf** (Pebesma 2018), the ease of use, reproducibility, and replicability of analysis is often enhanced when algorithms are implemented as part of a dedicated package. The **motif** package (Nowosad 2021) is the only example the authors could find of a package that is fully dedicated to landscape stratification in R using geospatial data. Although the methods offered by **motif** are effective for large-scale studies (Jasiewicz, Netzel, and Stepinski 2015; Nowosad 2021), their application is currently limited to rasters of categorical data. Thus, **motif** is not practical for the modeling of spatially continuous environmental phenomena, which is often a goal of landscape stratification routines.

This work presents the **rassta** package as a collection of algorithms for the spatial stratification of landscapes, sampling, and modeling of environmental phenomena. The **rassta** package is not intended as a drop-in replacement for statistically-robust environmental modeling approaches. Rather, it is intended to serve as a generalized framework to derive geospatial information that can be used to improve inference with these statistical approaches.

2 Conceptual overview and functionality

The algorithms in the **rassta** package assist in the analysis of environmental information related to the spatial variability of natural phenomena across landscapes. These functions focus on integrating standard geospatial techniques and quantitative analysis in a generalized framework for landscape stratification, sampling, and modeling. All of the functions in the **rassta** package take geospatial data in raster format as input. In the context of geographic information systems (GIS), the raster format can be considered a graphical representation of a matrix that is organized in rows and columns, and which may be stacked in multiple layers (e.g., multi-band satellite imagery). Each cell (pixel) in the raster contains a value representing a spatially-varying phenomenon, such as elevation or precipitation. A few functions in **rassta** also produce geospatial data in vector format. Vector data represents geometric entities in the form of points, lines, and polygons. The **rassta** package uses the highly efficient **terra** package as the backbone for handling raster and vector data. Most of the geospatial data manipulation with **terra** is performed in C++ and is based on two main R data types (classes): `SpatRaster` and `SpatVector`. Note that **terra** imports the **Rcpp** package (Eddelbuettel and François 2011) since **terra** uses C++ (including external pointers) to manipulate these classes.

Most of the functions implemented in **rassta** are interrelated in the sense that the outputs from some functions can be used as the inputs for others. This functional interrelation allows for a generalized framework to conduct spatial stratification, sampling, and modeling in a single package following a project-oriented approach. In general, the functions of **rassta** can be grouped into five categories: (a) landscape stratification; (b) landscape correspondence metrics; (c) stratified sampling; (d) spatial modeling; and (e) miscellaneous (Figure 1). Each category and its corresponding functions (except for miscellaneous) are theoretically founded on several studies focused on understanding spatially-varying natural phenomena across landscapes. In the next sections, the rationale behind each category and its functions is described. This description is complemented with references to corresponding scientific literature and includes code examples showing the application of each function with extensive use of plotting functions (for visualization purposes only). Most of the plotting functions are derived from the **terra** package using the `SpatRaster` and `SpatVector` classes. [Note: To reduce the extension of code examples, all the map and graph plotting functions were consolidated in the function `figure()`].

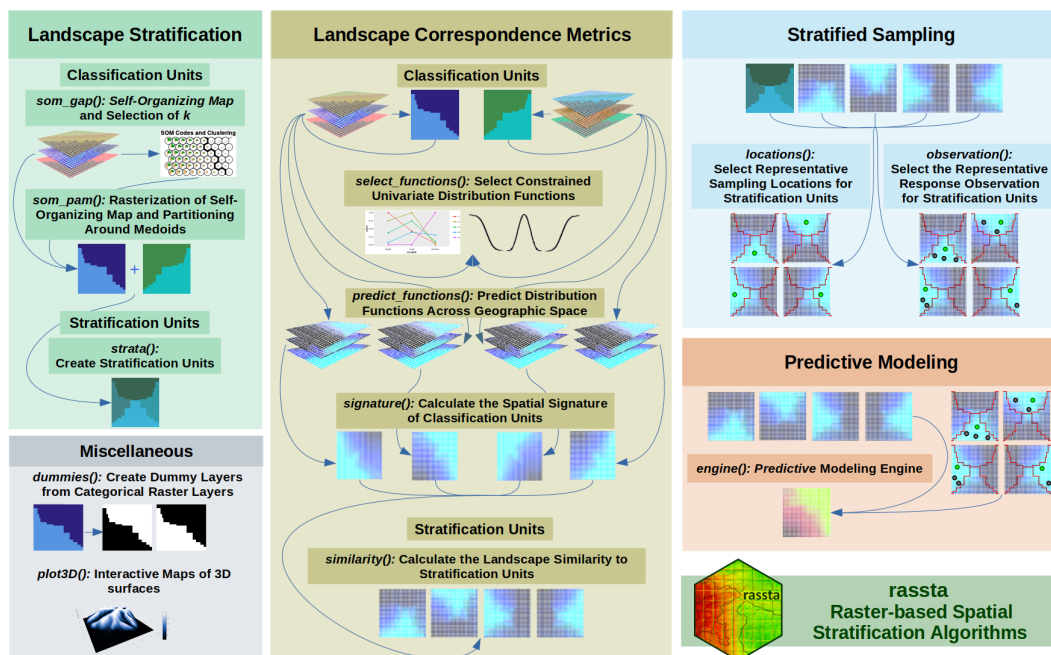


Figure 1: Functions of the **rassta** package. Connectors relate the inputs and outputs of the functions. The functions can be grouped in five categories: landscape stratification, landscape correspondence metrics, stratified sampling, predictive modeling, and miscellaneous.

Landscape stratification

Several studies have suggested the need to account for the hierarchical and multi-scale nature of landscape processes. Allen and Starr (1982) suggested that landscape processes can be explained through hierarchical multivariate structures given their multiple spatial and temporal scales. Based on

Dokuchaev's theory of soil formation (Glinka 1927) and the soil-landscape paradigm (Hudson 1992), McSweeney et al. (1994) proposed a nested model of soil-landscape processes at the physiographic, geomorphometric, and within-soil domains. Flügel (1995) suggested that the regionalization of hydrology-related processes should consider the multi-scale landscape heterogeneity in terms of soil, topography, geology, climate, and vegetation. These ideas have led these and other authors to formulate frameworks for the creation of spatial entities that stratify the landscape. The general purpose of these entities has been to define spatially-explicit domains that represent distinctive landscape processes and/or interactions (McSweeney et al. 1994). Accordingly, spatial stratification using *rassta* focuses on the creation of such domains (hereafter referred to as units).

The landscape stratification process with *rassta* follows a hierarchical approach similar to Austin and Heyligers (1989), who individually classified gradients of precipitation and elevation into intervals that were intersected with geologic classes for sampling purposes. Similarly, in *rassta*, a set of first-level units is created separately for each landscape factor under analysis. Then, multiple sets of first-level units are integrated into a single set of second-level units. The first-level units, called *classification units*, can be created outside of *rassta* via multicriteria analysis, statistical learning, or other methods. Moreover, the classification units can be formally defined through classification schemes, such as those based on taxonomic keys. The second-level units, called *stratification units*, result from the spatial intersection of multiple sets of classification units. Note that both classification and stratification units represent a spatial stratification for a given landscape. Figure 2 shows an example of a simple landscape stratification process based on two landscape factors, each with three raster layers representing continuous variables.

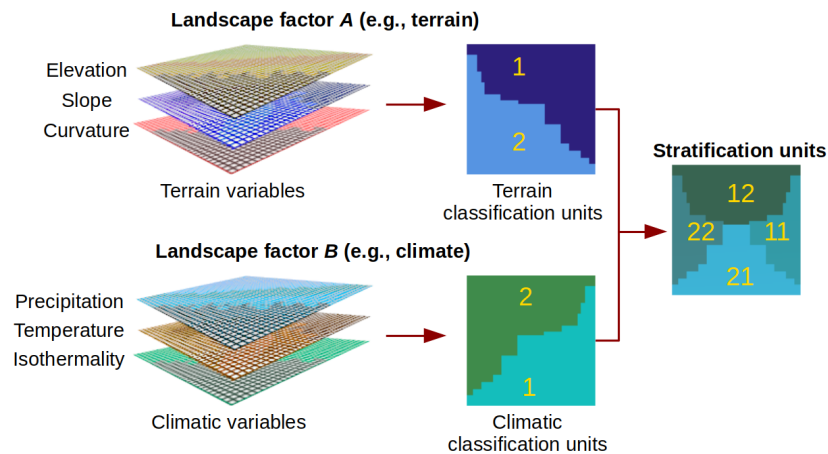


Figure 2: Schematic of a landscape stratification process. Raster layers of variables related to landscape factors are the inputs. The outputs are raster layers representing classification and stratification units.

There are three important aspects of the stratification approach used within *rassta* that must be considered. (a) One can simply create stratification units by incorporating variables from multiple landscape factors in a single classification process. However, the interpretation of results is often compromised when using a large number of variables in “all-in-the-bag” statistically driven classification schemes. (b) Multiple sets of classification units can belong to a single landscape factor, and each set can be created from variables at a distinct spatial scale. Presumably, this can account for the multi-scale nature of landscape factors in the stratification process. (c) A landscape factor can be represented by a single categorical variable, as in the case of geologic units or soil parent material. In this case, the landscape factor/variable is already in the form of classification units. Figure 3 shows a landscape stratification scenario like that addressed in (b) and (c).

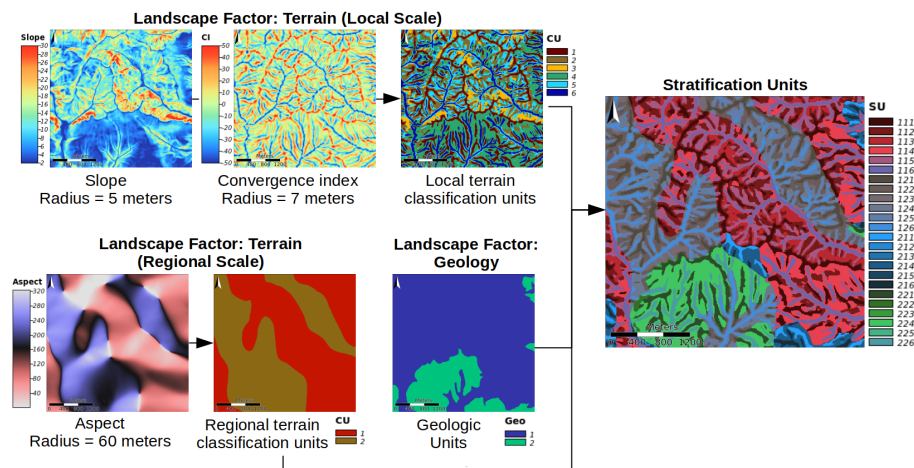


Figure 3: Schematic of a multi-scale landscape stratification process including a categorical variable. The stratification is based on three landscape factors: local scale terrain, regional scale terrain, and geology. Each terrain landscape factor is represented by raster layers of variables (local scale: slope and convergence index and regional scale: aspect and regional terrain). Geology is represented by a single categorical raster layer. Three sets of classification units (CU), one each for local terrain, regional terrain, and geology, are intersected to produce one set of stratification units (SU).

Classification units

A set of n classification units represents n distinct landscape configurations related to a single landscape factor. Note that the term *landscape configuration* is used here as a generic term for a particular pattern in the spatial variability of one or multiple variables belonging to a landscape factor. Currently, **rassta** allows the creation of classification units via unsupervised learning thanks to its functions `som_gap()` and `som_pam()`. The function `som_gap()` performs dimension reduction based on the self-organizing map (SOM) proposed by Kohonen (1990). The R package **kohonen** (Wehrens and Kruisselbrink 2018) is called internally by `som_gap()` to produce the SOM. The function `som_pam()` also performs cluster analysis on the SOM codes based on the partitioning around medoids (PAM) (Kaufman and Rousseeuw 1990), with estimation of the optimum number of clusters (k) through the gap statistic (Tibshirani, Walther, and Hastie 2001). It is important to mention that the output SOM object returned by `som_gap()` can be used as input for any other clustering algorithm (e.g., hierarchical, spectral, etc.) or statistical analysis outside of **rassta**.

The code below shows how `som_gap()` reduces the feature space and selects k clusters from four terrain variables. Note that the processing time of `som_gap()` is significant (around 162 seconds on a 4-cores Intel processor at 3.2 GHz for the following example). The processing time increases as the number of cells/layers in the argument `var.rast` increases, and/or as the argument `K.max` increases.

```
# Load the rassta and terra packages
library(rassta)
library(terra)
# Note that terra imports Rcpp, but if Rcpp is not automatically loaded then:
library(Rcpp)
# Get the data required to run the examples from rassta's installation folder
wasoil <- system.file("exdat/wasoil.zip", package = "rassta")
# Copy data to current working directory and extract files
file.copy(from = wasoil, to = getwd())
unzip("wasoil.zip")

# Set seed
set.seed(963)
# Multi-layer SpatRaster with 4 terrain variables
terr.var <- rast(c("height.tif", "midslope.tif", "slope.tif", "wetness.tif"))
# Scale variables to mean = 0 and standard deviation = 1
terr.varscale <- scale(terr.var)
# Dimensionality reduction and estimation of optimum k (max k to evaluate: 12)
terr.som <- som_gap(terr.varscale, xdim = 10, ydim = 10, K.max = 12)
# Plot results
```

```
figure(4, d = list(terr.var, terr.som))
```

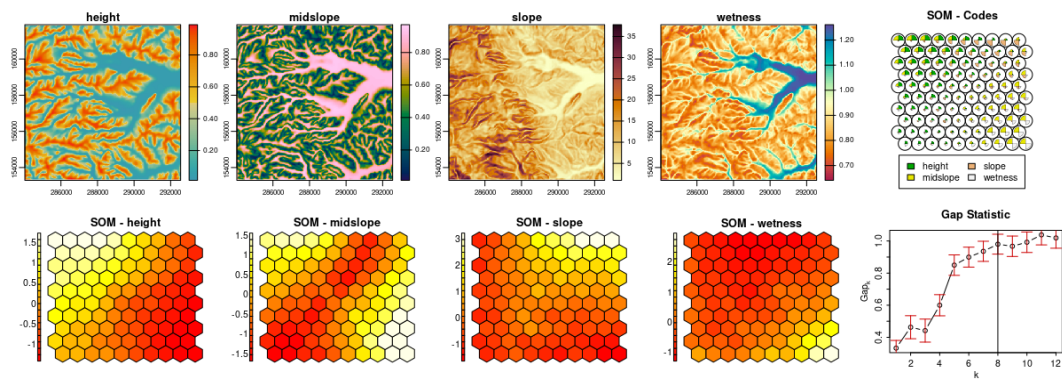


Figure 4: Dimension reduction and selection of number of clusters (k). The top row shows four terrain variables (height, midslope, slope, and wetness) that are used to generate the self-organizing map (SOM). The bottom row shows the reduced feature space of each variable and the Gap statistic that is used to select k for the construction of classification units.

The function `som_pam()` creates raster versions from the outputs of `som_gap()`. The code below shows how `som_pam()` creates raster versions of the SOM grid and PAM clustering computed in the previous example.

```
# Rasterization of terrain SOM grid and terrain PAM clustering
terr.sompam <- som_pam(ref.rast = terr.var[[1]], kohsom = terr.som$SOM,
                      k = terr.som$Kopt)

# Plot results
figure(5, d = list(terr.sompam, terr.var))
```

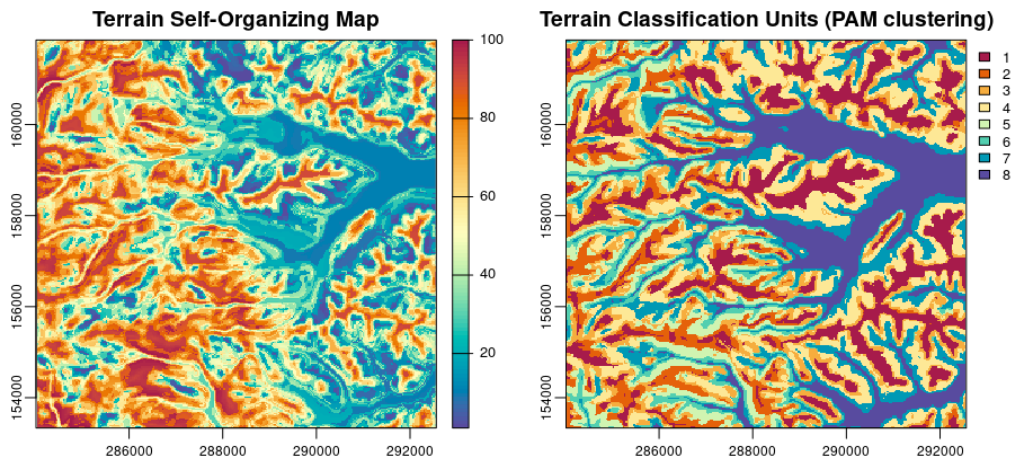


Figure 5: SOM grid and PAM clustering. Rasterized versions of the terrain SOM grid (left) and the terrain PAM clustering (right) are produced. The resulting clusters represent the classification units for the terrain landscape factor.

Note that the approach for creating classification units should not be limited to that offered by `som_gap()` and `som_pam()`. There are many other approaches outside of `rassta` that can be followed, such as supervised classification based on statistical learning, or GIS-based multicriteria analysis. The best approach may depend on the research question(s) being addressed. Therefore, the selection of the proper approach and the optional use of other R packages and/or GIS software is left to the user. Also, note that classification units created outside of `rassta` are completely compatible with `rassta` objects and methods if the units are represented through the `SpatRaster` class from `terra`.

Stratification units

A set of n stratification units represents n distinct landscape configurations related to multiple landscape factors. Note that the term *landscape configuration* is used here as a generic term for a particular pattern in the spatial variability of multiple variables belonging to multiple landscape factors, or to the same factor represented at multiple spatial scales. The function `strata()` allows the spatial intersection of multiple sets of classification units into a single set of stratification units. This function also assigns a unique numeric code to each stratification unit. The numeric code makes it possible to trace back each classification unit composing a given stratification unit. The code below shows the construction of stratification units with `strata()` using classification units from three landscape factors (climate, soil parent material, and terrain).

```
# Multi-layer SpatRaster with 3 sets of classification units
all.cu <- rast(c("climate.tif", "material.tif", "terrain.tif"))
# Stratification units
su <- strata(cu.rast = all.cu)
# Plot results
figure(6, d = list(su, all.cu))
```

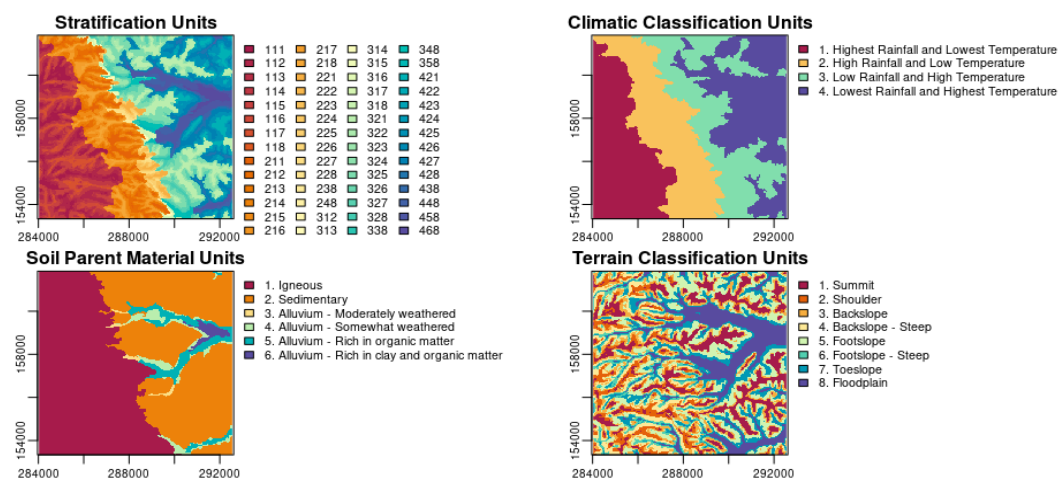


Figure 6: Creation of stratification units from sets of classification units. A set of classification units is produced for each of three landscape factors: climate, soil parent material, and terrain. The spatial intersection of these sets results in the stratification units for the landscape (upper left map).

Metrics of landscape correspondence

There are two metrics of landscape correspondence that can be calculated with `rassta`: (a) the spatial signature of classification units, and (b) the landscape similarity to stratification units. These metrics quantify the relative correspondence between any location across geographic space and landscape configurations represented by classification and stratification units. Several studies have applied similar concepts related to continuous correspondence between landscape configurations for the modeling of spatially-varying phenomena. Early examples include studies using multivariate distance metrics in the feature space for SDM (Carpenter, Gillison, and Winter 1993) and studies applying the fuzzy set theory (Zadeh 1965) for multicriteria evaluation (Burrough 1989), DSM (Zhu and Band 1994) and landform classification (MacMillan et al. 2000).

Spatial signature of classification units

The spatial patterns of the degree of correspondence between any landscape configuration and the configuration represented by a given classification unit are defined as the *spatial signature*. The spatial signature is represented by a raster layer of continuous values that results from the cell-wise aggregation of empirical distribution functions mapped over geographic space. Each distribution function corresponds to one variable and relates the classification unit to “typical” values of the variable within the classification unit. The concept of spatial signature is based on the work of Pike and Rozema (1975) and Pike (1988). These authors used the term *geometric signature* to describe a set of

sample statistics (e.g., mean, standard deviation) of terrain variables (e.g., slope, curvature) used to distinguish “geomorphically disparate landscapes” (Pike 1988).

The spatial signature in **rassta** replaces the geometric signature’s measurements of central tendency and dispersion statistics with statistical distribution functions generated across geographic space. The statistical distribution functions used in **rassta** are: (a) the probability density function (PDF) based on the kernel density estimation, (b) the empirical cumulative distribution function (ECDF), and (c) an inverted version of the ECDF (iECDF). Note that the spatial signature concept is somewhat similar to the virtual ecological niche (Hirzel, Helfer, and Metral 2001) and the multivariate environmental similarity surface (Elith, Kearney, and Phillips 2010), which are implemented in R through the packages **virtualespecies** (Leroy et al. 2016) and **dismo** (Hijmans et al. 2020), respectively. Figure 7 and Figure 8 show an illustration and a pseudocode of the process to calculate the spatial signature of a classification unit, respectively. Note that the function FUNSIG() in the pseudocode is just a placeholder to encompass the three functions from **rassta** that are required to calculate spatial signatures. These functions are `select_functions()`, `predict_functions()`, and `signature()`, each will be further discussed next.

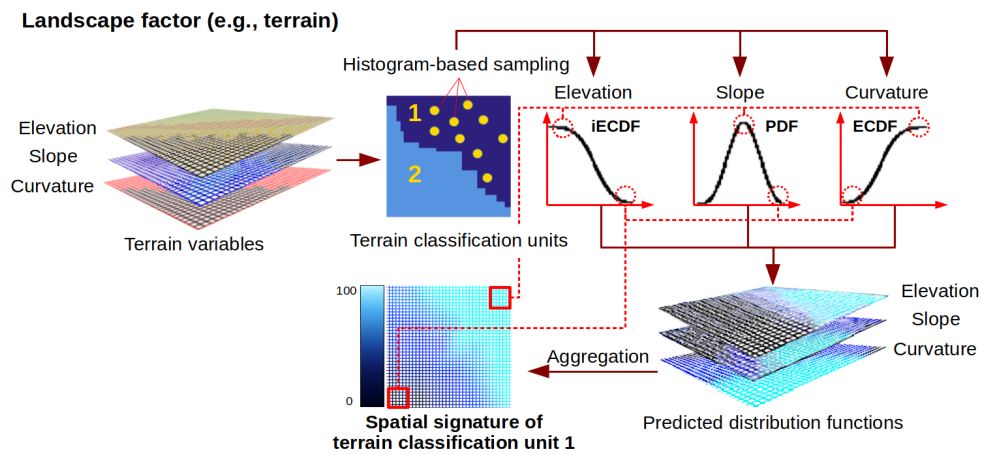


Figure 7: Schematic of the calculation process for spatial signatures. A set of classification units is produced using three variables. A distribution function is calculated for each variable within classification unit 1, and then predicted across geographic space. The predicted functions for unit 1 are aggregated, which results in the spatial signature of that unit.

 Select and predict distribution functions, then calculate spatial signature

Require:

```

   $V_{v1, \dots, vn}$ , multi-layer SpatRaster with  $n$  layers (i.e., variables)
   $C_{c1, \dots, cn}$ , single-layer SpatRaster representing  $n$  classification units
1: function FUNSIG( $V_{v1, \dots, vn}, C_{c1, \dots, cn}$ )
2:   for each  $v$  in  $V_{v1, \dots, vn}$  do
3:      $S_{c1, \dots, cn} \leftarrow$  SUMMARY( $v, C_{c1, \dots, cn}$ )  $\triangleright$  within-unit statistic (e.g., mean) of  $v$ 
4:     for each  $c$  in  $C_{c1, \dots, cn}$  do
5:        $v_c \leftarrow$  EXTRACT( $v, c$ )  $\triangleright$  Extract observations of  $v$  within  $c$ 
6:       if  $S_c = \text{MAX}(S_{c1, \dots, cn})$  or  $S_c = \text{MIN}(S_{c1, \dots, cn})$  then  $\triangleright S_c$ : statistic of  $v$  within  $c$ 
7:          $f \leftarrow$  ECDF  $\triangleright$  select distribution function for  $v_c$ 
8:          $v_{cs} \leftarrow$  QUANTILES( $v_c$ )  $\triangleright$  sample from  $v_c$  to reduce processing time
9:          $c_f \leftarrow f(v_{cs})$   $\triangleright$  calculate distribution function of  $v_{cs}$ 
10:        if  $S_c = \text{MIN}(S_{c1, \dots, cn})$  then
11:           $c_f \leftarrow$  INVERT( $c_f$ )
12:        end if
13:      else
14:         $f \leftarrow$  PDF
15:         $v_{cs} \leftarrow$  CENTRALPOINTS( histogram( $v_c$ ) )
16:         $c_f \leftarrow f(v_{cs})$ 
17:      end if
18:       $c_{loess} \leftarrow$  LOESS( $y \sim x$ );  $y = c_f, x = v_{cs}$   $\triangleright$  fit loess regression
19:       $C_{v_{predfun}} \leftarrow$  PREDICT( $c_{loess}, v$ )  $\triangleright$  predict the distribution function of  $v_{cs}$  across  $v$ 
20:    end for
21:  end for
22:  for each  $c$  in  $C$  do
23:     $C_{predfun} \leftarrow$   $C_{v1_{predfun}}, \dots, C_{vn_{predfun}}$   $\triangleright$  Predicted distribution functions for  $c$  (one per each  $v$ )
24:     $C_{sig} \leftarrow$  AGGREGATE( $C_{predfun}$ )  $\triangleright$  calculate Spatial signature of  $c$ 
25:  end for
26: end function

```

Figure 8: Pseudocode of the calculation process for spatial signatures. The calculation process involves the selection, prediction, and aggregation of distribution functions. The spatial signature is calculated for each classification unit in a set.

An important assumption is made when using the PDF, ECDF, and iECDF to characterize the typical values of a given variable within a given classification unit. The position of a value within the distribution function is an indicator of how typical the value is in terms of the variable's distribution within the classification unit. For instance, values closer to, or at the peak of the PDF are assumed to be the most typical values of the variable within the classification unit. Contrarily, values at the tails of the PDF are the less typical. Although one could simply use the PDF as a generalized function to denote typical values, this function assigns the same weight to values at the tails of the distribution regardless of the tail's location (left or right). In some cases, *a priori* knowledge can dictate that typical values of a variable within a given classification unit are those approaching $+\infty$, or those approaching $-\infty$. The use of the ECDF and the iECDF is intended for those cases. More specifically, if a classification unit is known to be associated with a variable's extreme values toward $+\infty$, then the ECDF can be used to represent this association. Conversely, if the classification unit is associated with those variable's extreme values toward $-\infty$, then the iECDF can be used.

The function `select_functions()` allows the user to select the statistical distribution function used to represent the typical values for a given variable within a specific classification unit. Both automatic and interactive selection modes are supported, with the latter based on a [shiny](#) app (Chang et al. 2021). The automatic selection of distribution functions is based on within-unit statistics, also referred to as *zonal statistics* in the GIS literature, and it follows the criteria described next:

- PDF = when the mean (or median) of the variable's values within the classification unit is neither the maximum nor the minimum of all the mean (or median) values across all the units.
- ECDF = when the mean (or median) of the variable's values within the classification unit is the maximum of all the mean (or median) values across all the units.
- iECDF = when the mean (or median) of the variable's values within the classification unit is the minimum of all the mean (or median) values across all the units.

The code below shows the automatic selection of statistical distribution functions for four climatic classification units and two variables with `select_functions()`.

```

# Multi-layer SpatRaster with 2 climatic variables
clim.var <- rast(c("precipitation.tif", "temperature.tif"))
# Single-layer SpatRaster with 4 climatic classification units

```

```

clim.cu <- rast("climate.tif")
# Automatic selection of statistical distribution functions
clim.difun <- select_functions(cu.rast = clim.cu,
                              var.rast = clim.var,
                              mode = "auto")

# Plot results
figure(8, d = list(clim.difun, clim.cu, clim.var))

```

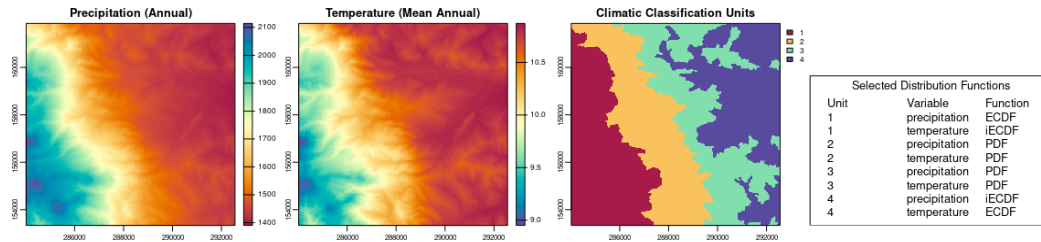


Figure 9: Selection of distribution functions. A set of four climatic classification units are produced using two variables: precipitation and temperature. A distribution function is selected for each variable within each classification unit.

The selected distribution functions can be used to generate predictions of distribution function values over geographic space with the function `predict_functions()` as shown in the code below. The predictions are generated by fitting a locally estimated scatterplot smoothing (LOESS) regression with the within-unit distribution function's values (y) and the within-unit variable's values (x). The fitted LOESS and the raster layer of the variable are then used to predict new distribution function values across geographic space.

```

# Multi-layer SpatRaster of climatic variables and classification units
clim.all <- c(clim.var, clim.cu)
# Output table from select_functions()
df <- clim.difun$distfun
# Predicted distribution functions for climatic variables
clim.pdfif <- predict_functions(cuvar.rast = clim.all,
                              cu.ind = 3,
                              cu = df$Class.Unit,
                              vars = df$Variable,
                              dif = df$Dist.Func)

# Plot results
figure(9, d = list(clim.pdfif, clim.cu))

```

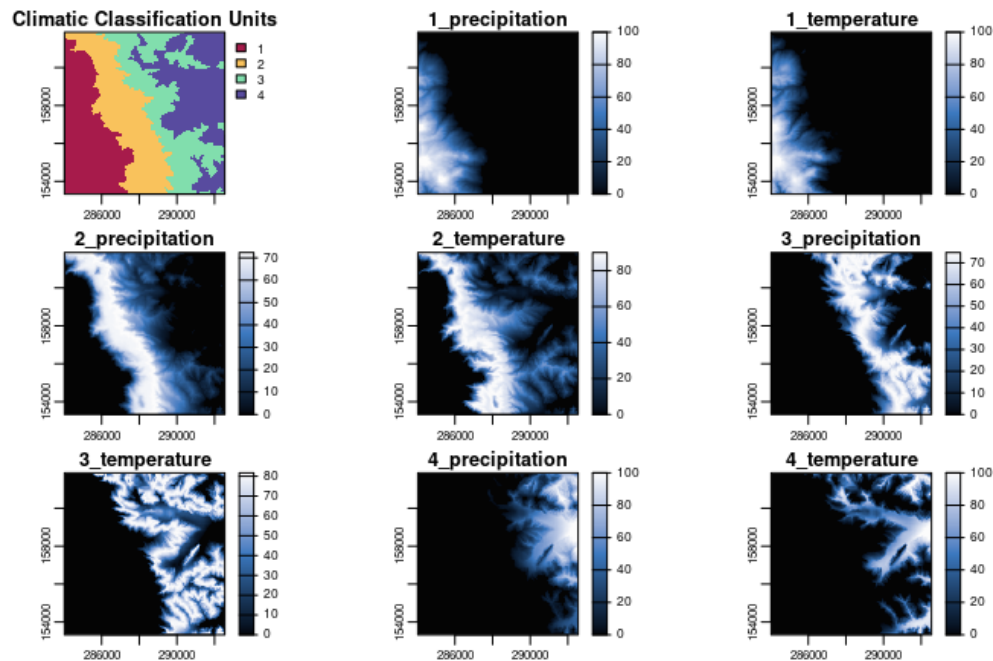



Figure 10: Prediction of distribution functions. A selected distribution function for each variable is predicted across geographic space. The predicted distribution function relates the landscape to a classification unit with regard to a variable.

The function `signature()` calculates the spatial signature of a given classification unit by aggregating all of the predicted distribution functions associated with the unit. The code below shows the calculation of spatial signatures with `signature()`. Note that the arguments `inprex` and `outname` allow the user to identify the raster layers representing the predicted distribution functions associated with each classification unit in a set, and to assign a unique name to each resulting raster layer of spatial signature, respectively.

```
# Spatial signatures from distribution functions predicted for climatic variables
clim.sig <- signature(pdif.rast = clim.pdif,
                    inprex = paste(seq(1, 4), "_", sep = ""),
                    outname = paste("climate_", seq(1, 4), sep = ""))

# Plot results
figure(10, d = list(clim.sig, clim.cu))
```

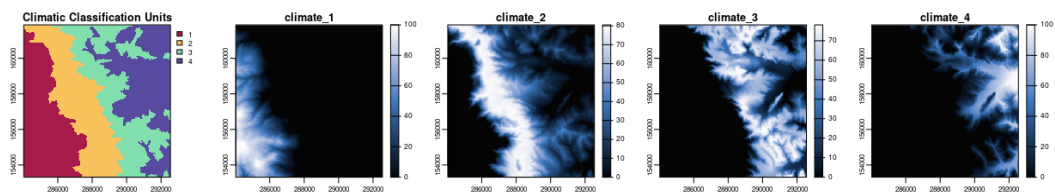


Figure 11: Calculation of spatial signatures. For each climatic classification unit (1 thru 4), the distribution functions (see Figure 10) are aggregated (e.g., mean pixel value) to produce the spatial signature of the unit. The spatial signature relates each position in the landscape to the landscape configuration represented by a classification unit.

Landscape similarity to stratification units

The spatial patterns of the degree of correspondence between any landscape configuration and the landscape configuration represented by a given stratification unit are defined as the *landscape similarity*. The landscape similarity is represented by a raster layer of continuous values, which results from the cell-wise aggregation of the spatial signatures of multiple classification units. This aggregation is possible because any given stratification unit is the result of the spatial intersection of multiple classification units, commonly one per landscape factor or factor scale (see Figure 2 and 3). Moreover,

each classification unit has one spatial signature associated with it. Therefore, any given stratification unit will be associated with multiple spatial signatures, which can be cell-wise aggregated to calculate the landscape similarity. Figure 12 shows an example of the calculation process for a layer of landscape similarity to stratification unit.

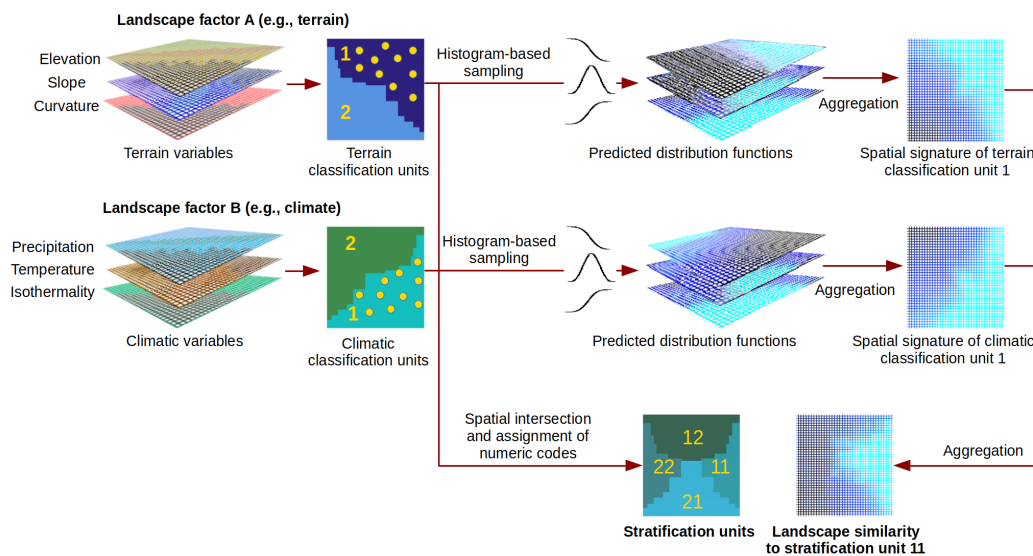


Figure 12: Schematic of the calculation process for landscape similarities. Sets of variables for each landscape factor (terrain and climate) are combined to produce sets of classification units (two each for terrain and climate), which are further combined to produce stratification units (12, 11, 21, and 22). Thus, each stratification unit has two classification units associated with it. Moreover, each classification unit has a spatial signature associated with it. Aggregating the spatial signatures of classification unit 1 for climate and unit 1 for terrain, both associated with stratification unit 11, results in the landscape similarity to that stratification unit.

The function `similarity()` calculates the landscape similarity layer for each stratification units in a given set (with the set being represented by a single-layer `SpatRaster` object), as shown in the following example. The argument `su.code` indicates the name of the landscape factors/factor scales used to create the stratification units, and the digit position (start, end) of the classification units' ID in the stratification unit's numeric code.

```
# Multi-layer SpatRaster with spatial signatures of classification units
clim.sig <- rast(list.files(pattern = "climate_")) # For climatic units
mat.sig <- rast(list.files(pattern = "material_")) # For soil parent material units
terr.sig <- rast(list.files(pattern = "terrain_")) # For terrain units
# Single-layer SpatRaster of stratification units
su <- rast("su.tif")
# Landscape similarity to stratification units
su.ls <- similarity(su.rast = su, sig.rast = c(clim.sig, mat.sig, terr.sig),
  su.code = list(climate = c(1, 1), material = c(2, 2),
    terrain = c(3, 3)))
# Plot results
figure(12, d = list(su.ls, su, clim.sig, mat.sig, terr.sig))
```

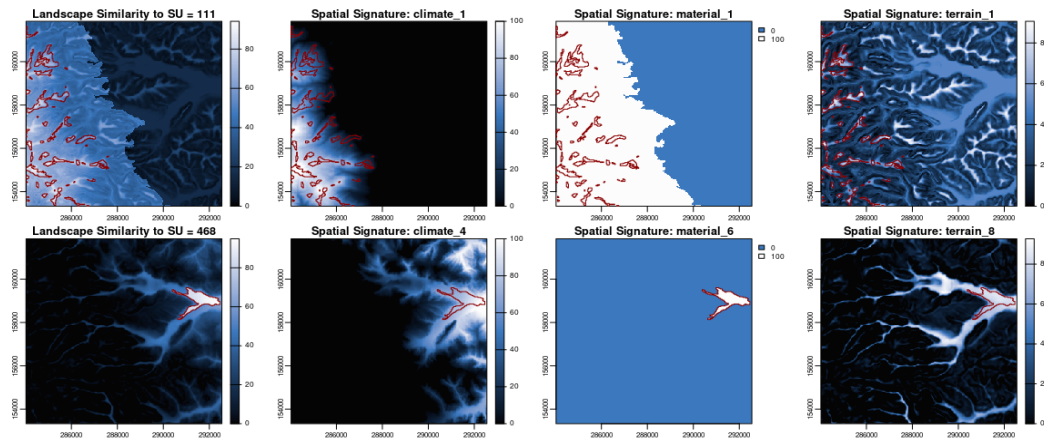


Figure 13: Metrics of landscape correspondence. Landscape similarity (extreme left) and spatial signatures for climate, parent material (material), and terrain associated with stratification units (SU) 111 (top row) and 468 (bottom row). The red polygons indicate the boundaries of the corresponding SU defined through the aggregation (i.e., mean pixel value) of the set of spatial signatures for that SU.

Stratified non-probability sampling

Stratified sampling is an efficient technique for achieving an adequate representation of environmental variability, reducing cost of field work, and improving modeling with limited observations (Austin and Heyligers 1989; Wessels et al. 1998; Guisan and Zimmermann 2000; Zhu et al. 2008; West et al. 2016). Accordingly, sampling with *rassta* to select observations/sampling locations is performed in a stratified fashion using stratification units. Additionally, the raster layers of landscape similarity to stratification units can be included in the sampling process. Including the landscape similarity layers results in a non-probability sample. For each stratification unit, the sampling process selects the observation(s)/sampling location(s) at the raster cell where the highest landscape similarity value occurs, resulting in a stratified, non-probability sample that is biased towards maximizing the representativeness of landscape configurations. This idea of biased, stratified sampling is based on the work of Gillison (1983); Gillison and Brewer (1985), Austin and Heyligers (1989), and Zhu et al. (2008). These authors have suggested that bias related to landscape configurations is relevant for the maximization of environmental representativeness, detection of maximum diversity, and representation of non-stochastic control-response relationships.

The function `observation()` performs the automatic selection of the representative response observation for each stratification unit in a given set. Given a stratification unit, the unit's representative response observation is that whose value best reflects the influence that the unit's landscape configuration exerts on the response. This function requires a set of observations/samples already collected for a set of stratification units. Currently, `observation()` selects observations based on the following methods: (a) *mls*: select the observation at the raster cell with the maximum landscape similarity value; (b) *mrv*: select the observation whose response value is the median of all the values; and (c) *random*: select an observation at random. Note that the latter represents a case of stratified random sampling.

The code below shows the selection of representative soil organic carbon (SOC) observations based on the maximum landscape similarity method. Note that the arguments `su.rast` and `ls.rast` require the stratification units and landscape similarity layers previously created with `strata()` and `similarity()`, respectively.

```
# SpatVector with SOC observations for stratification units
soc.obs <- vect("soc.shp")
# Representative SOC observation for each stratification unit
su.obs <- observation(su.rast = su, obs = soc.obs, col.id = 1, col.resp = 2,
  method = "mls", ls.rast = su.ls$landsim)
# Plot results
figure(13, d = list(su.obs, soc.obs, su))
```

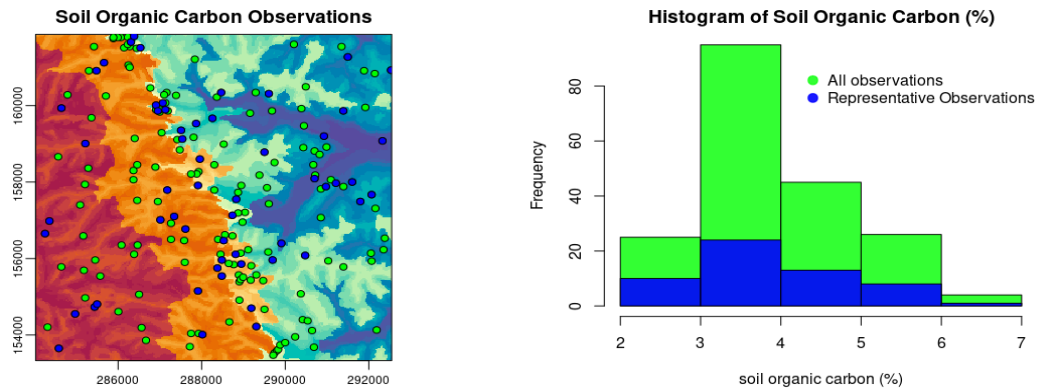


Figure 14: Selection of representative observations. Green points in the map represent the complete set of observations. Blue points represent the representative observation for each stratification unit.

The function `locations()` performs the automatic selection of the representative sampling location(s) for each stratification unit in a given set, where the representative sampling location is the raster cell where the highest landscape similarity value occurs. Currently, `locations()` implements two selection methods: (a) *buffer*: select sampling locations within areas with landscape similarity values above a certain threshold; and (b) *absolute*: select sampling locations with the highest landscape similarity values. The code below shows the use of `locations()` based on the buffer method.

```
# Representative sampling location and its buffer area for each stratification unit
su.samp <- locations(ls.rast = su.ls$landsim, su.rast = su, method = "buffer")
# Plot results
figure(14, d = list(su.samp, su))
```

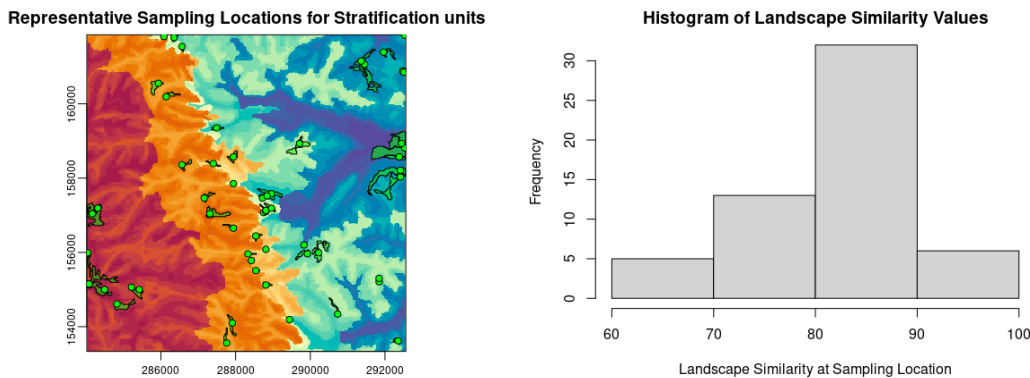


Figure 15: Selection of representative sampling locations. Green points in the map represent the sampling location for each stratification unit. Green polygons represent the buffer area for each sampling location.

Predictive modeling

Predictive modeling with `rassta` is based on the assumption that each stratification unit represents a distinct landscape configuration and that this configuration influences a natural phenomenon in a distinctive manner. It is assumed that the influence that a stratification unit's landscape configuration has on response phenomena at a specific location (i.e. raster cell) is proportional to the unit's landscape similarity value at that raster cell. Therefore, given a stratification unit x , the corresponding raster layer of landscape similarity ls , the response y , and a raster cell c , the greater the value of ls at c , the more similar y at c will be to the typical y for x . The typical (i.e., representative) value of a response phenomenon for a given stratification unit can be defined in several ways. For instance, if a response phenomenon was sampled/measured multiple times within a given stratification unit, the typical response value could be that from the sample/measurement at the raster cell with the highest landscape similarity value (see `observation()`).

Several studies have used landscape similarity layers to model the spatial variability of natural phenomena. These studies argue that the use of similarity layers is appropriate in cases when (a)

available observations for modeling are limited (Zhu et al. 2008); (b) initial spatial distribution patterns are needed for survey design (Carpenter, Gillison, and Winter 1993); (c) expert-driven selection of informative variables is possible (Knick and Dyer 1997); (d) *a priori* knowledge of response-control relationships in the form of conceptual models is available (Zhu et al. 2010; Schmidt, Tonkin, and Hewitt 2005); and (e) discriminating between (ecologically) positive and negative deviations from reference environments is required (Watrous et al. 2006). Accordingly, `engine()` allows the modeling of environmental phenomena with a number of training observations as few as the number of landscape similarity layers [cases (a) and (b)]; training observations and landscape similarity layers as outcomes of expert-driven landscape stratification [cases (c) and (d)]; and landscape similarity layers derived from spatial signatures that discriminate between the tails of distribution functions [case (e)].

Modeling with `rassta` is performed using the function `engine()`. For continuous responses, `engine()` performs a weighted average involving representative response values and landscape similarity layers. For a raster cell c , the modeled response value is equal to the weighted average of the representative values for those stratification units with the highest landscape similarity values at c . The stratification units with the highest landscape similarity values at c can be considered as the *nearest neighbors* (in feature space) of the landscape configuration at c . These nearest neighbors are called *winning stratification units*, and the weight of their corresponding representative value is proportional to the winning unit's landscape similarity value at c . For categorical responses, the modal response value of the winning stratification units replaces the weighted average. Figure 16 shows an example of the modeling process for continuous responses with `rassta`.

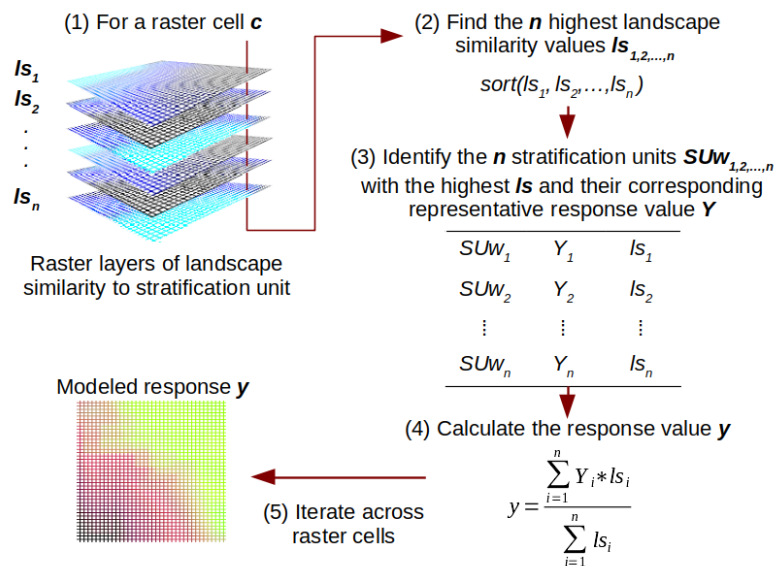


Figure 16: Schematic of the modeling process with `rassta`. The modeling process is performed in a cell-wise fashion. The inputs required are the raster layers of landscape similarity and the representative observations for each stratification unit.

Note that the weighted average for modeling phenomena across geographic space has been widely applied in GIS-based multicriteria decision analysis (GIS-MCDA). In GIS-MCDA, attributes (i.e., variables) in the form of raster layers are weighted according to expert criteria. The weighted variables are then combined through (cell-wise) overlay operators such as multiplication, addition and (ordered) averaging. The resulting value at each cell represents the relative suitability for a certain condition/decision (Malczewski 2006). The function `engine()` generalizes the weighted overlay process of GIS-MCDA by allowing the use of sampled/measured data of a response phenomenon in conjunction with the landscape similarity layers acting as weighted variables. This generalization allows the modeling of real-valued phenomena in continuous or categorical form. The modeling approach of `engine()` is almost the same as that proposed by Zhu (1997) to model landscape attributes across geographic space. The difference between `engine()` and the approach of Zhu (1997) is that `engine()` allows the selection of the number of landscape similarity layers for the weighted average calculation. Presumably, restricting the number of layers will reduce the shortening ('shrinking') effect that weighted averaging has on the range of modeled continuous response values (Nolan et al. 2019).

The code below demonstrates the use of `engine()` for the predictive modeling of soil organic carbon. Note that the representative response values (argument `su.repobs`) are those previously selected with `observation()`, and that the layers of landscape similarity (argument `ls.rast`) are those

previously created with `similarity()`.

```
# Table with the numeric code of stratification units and representative SOC values
su.soc <- su.obs$su_repobs[, c("SU", "soc")]
# engine() requires a (tiled) SpatVector with the boundaries of the area of interest
aoi <- vect("aoi.shp")
# engine() writes results directly on disk
if (dir.exists("soc") == FALSE) {dir.create("soc")} # Create directory
# Spatial modeling of SOC across the landscape based on 3 winning stratification units
soc <- engine(ls.rast = su.ls$landsim, n.win = 3, su.repobs = su.soc,
             tiles = aoi, outdir = "soc", overwrite = TRUE)
figure(16, d = list(soc, "soc_valid.shp")) # Plot results
```

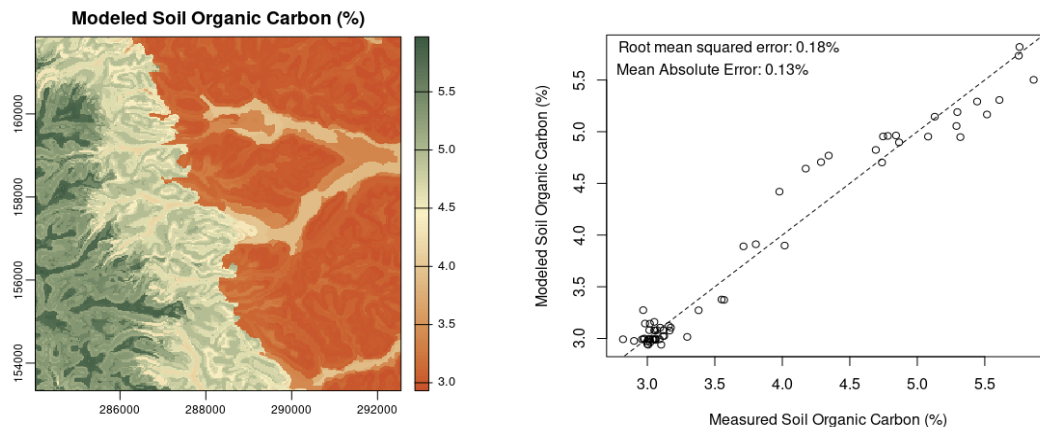


Figure 17: Modeled soil organic carbon (SOC) content (percent). The map shows the modeled SOC values across the landscape. The plot shows the modeled (y) versus the measured (x) SOC values based on 62 independent observations.

Miscellaneous

The spatial signature only applies to classification units created from continuous variables. Thus, spatial signatures cannot be calculated for classification units that represent categorical variables, such as land use/land cover. In such cases, a one-hot encoding can be applied to produce binary layers for the units. These layers are considered the spatial signatures of the classification units. The code below shows the creation of binary layers for soil parent material units with `dummies()`.

```
# Multi-layer SpatRaster of soil parent material units
mat.cu <- rast("material.tif")
# Binary layers for each soil parent material unit and their maps
mat.sig <- dummies(mat.cu, preval = 100, absval = 0)
figure(17, d = mat.sig) # Plot results
```

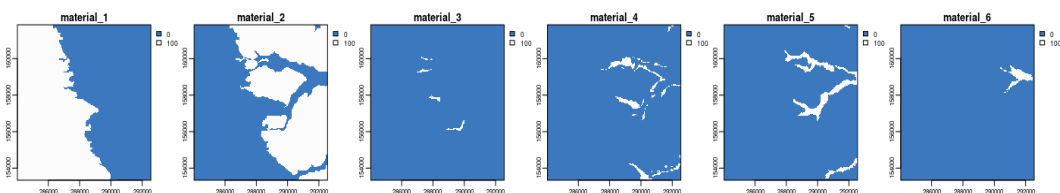


Figure 18: Construction of binary layers. Binary layers act as the spatial signatures for categorical variables. In this example, soil parent material acts as both landscape factor and classification units.

The function `plot3D()` produces interactive maps showing the 3-dimensional (XYZ) variability in raster layers representing continuous variables. The XYZ reference positions are obtained from a user-supplied elevation layer. For large raster layers (large spatial coverage and/or high spatial resolution), this function allows the option to decrease resolution and subset the data. The code below shows how `plot3D()` creates a 3D map for SOC, as modeled with `engine()`.

```
# Single-layer SpatRaster of terrain elevation and the 3D SOC map
elev <- rast("elevation.tif")
plot3D(c(elev, soc), z = 1, ex = 0.2, pals = "Fall", rev = TRUE) # 3D map
```

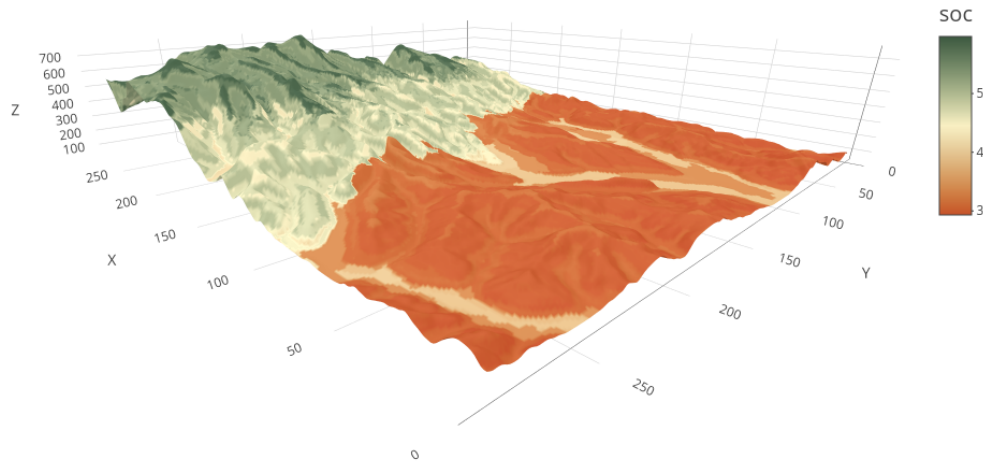


Figure 19: 3D map of SOC (percent). The Z dimension is obtained from a reference terrain model. Visit the online article to access the interactive version of the map.

3 Future versioning and summary

This work presented the **rassta** package for spatial stratification, sampling, and modeling of environmental phenomena within the R environment. Future versioning of the **rassta** package will focus on developing new approaches for spatial stratification. Stratification based on spatial intersection may not be feasible to implement in highly complex landscapes because these landscapes may require many (sets of) classification units to accurately represent the spatial variability of landscape factors, leading to over-stratification, and thus, greater demand for samples/observations to conduct predictive modeling based on landscape similarity. One plausible solution is the application of the stratification methods presented by Jasiewicz, Netzel, and Stepinski (2015), Jasiewicz, Stepinski, and Niesterowicz (2018), Nowosad (2021), and Nowosad and Stepinski (2021). However, these methods have been purposely designed for studies with continental/global applications. Therefore, these methods should be adapted for **rassta** to tailor their application at local scales to allow for more precise representations of natural phenomena and their spatial variability. Another focus of versioning can be new functions to visualize the variability of response phenomena relative to the hierarchical structure represented by the stratification units. Lastly, future versioning of **rassta** should also consider the user's experiences to ensure its general applicability.

The core ideas implemented in the **rassta** package include the multi-scale, hierarchical landscape stratification based on spatial intersection, the application of non-parametric distribution estimators to define the typical landscape configuration of stratification units, and the use of spatially explicit landscape correspondence metrics for non-probability sampling and predictive modeling. Some of these ideas have previously been implemented in R through a few packages dedicated to the analysis of geospatial data. Nevertheless, **rassta** offers a unified, generalized framework to conduct multiple landscape stratification routines through a dedicated set of algorithms. Moreover, spatially-explicit information created with **rassta**, like stratification units, landscape similarity layers, and representative observations, can be embedded into statistically robust modeling approaches to optimize the analysis of environmental phenomena.

References

- Allen, T. F. H., and B. Starr. 1982. *Hierarchy: Perspectives for Ecological Complexity*. University of Chicago Press. <https://doi.org/10.7208/chicago/9780226489711.001.0001>.
- Araújo, Miguel B., and Antoine Guisan. 2006. "Five (or so) Challenges for Species Distribution Modelling." *Journal of Biogeography* 33 (10): 1677–88. <https://doi.org/10.1111/j.1365-2699.2006.01584.x>.
- Arrouays, D., A. McBratney, J. Bouma, Z. Libohova, A. Richer-de-Forges, C. Morgan, P. Roudier, L. Poggio, and V. Mulder. 2020. "Impressions of Digital Soil Maps: The Good, the Not so Good, and

- Making Them Ever Better." *Geoderma Regional* 20: e00255. <https://doi.org/10.1016/j.geodrs.2020.e00255>.
- Austin, M. P., and P. C. Heyligers. 1989. "New Approach to Vegetation Survey Design: Gradsect Sampling." *Nature Conservation: Cost Effective Biological Surveys and Data Analysis* 5: 31–36.
- Baldwin, Doug, Kusum Naithani, and Henry Lin. 2017. "Combined Soil-Terrain Stratification for Characterizing Catchment-Scale Soil Moisture Variation." *Geoderma* 285: 260–69. <https://doi.org/10.1016/j.geoderma.2016.09.031>.
- Burrough, Peter. 1989. "Fuzzy Mathematical Methods for Soil Survey and Land Evaluation." *Journal of Soil Science* 40 (3): 477–92. <https://doi.org/10.1111/j.1365-2389.1989.tb01290.x>.
- Carpenter, Guy, A. N. Gillison, and J. Winter. 1993. "DOMAIN: A Flexible Modelling Procedure for Mapping Potential Distributions of Plants and Animals." *Biodiversity & Conservation* 2 (6): 667–80. <https://doi.org/10.1007/bf00051966>.
- Chang, W., J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. 2021. **shiny**: *Web Application Framework for r*. <https://CRAN.R-project.org/package=shiny>.
- Chuvieco, E., I. Aguado, M. Yebra, H. Nieto, J. Salas, M. Pilar Martín, L. Vilar, et al. 2010. "Development of a Framework for Fire Risk Assessment Using Remote Sensing and Geographic Information System Technologies." *Ecological Modelling* 221 (1): 46–58. <https://doi.org/10.1016/j.ecolmodel.2008.11.017>.
- Eddelbuettel, E., and R. François. 2011. "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software* 40 (8): 1–18. <https://10.18637/jss.v040.i08>.
- Elith, Jane, Michael Kearney, and Steven Phillips. 2010. "The Art of Modelling Range-Shifting Species." *Methods in Ecology and Evolution* 1 (4): 330–42. <https://doi.org/10.1111/j.2041-210x.2010.00036.x>.
- Elith, Jane, and John R. Leathwick. 2009. "Species Distribution Models: Ecological Explanation and Prediction Across Space and Time." *Annual Review of Ecology, Evolution, and Systematics* 40: 677–97. <https://doi.org/10.1146/annurev.ecolsys.110308.120159>.
- Flügel, Wolfgang-Albert. 1995. "Delineating Hydrological Response Units by Geographical Information System Analyses for Regional Hydrological Modelling Using PRMS/MMS in the Drainage Basin of the River Bröl, Germany." *Hydrological Processes* 9 (3-4): 423–36. <https://doi.org/10.1002/hyp.3360090313>.
- Gillison, Andrew. 1983. "Gradient Oriented Sampling for Resource Surveys - the Gradsect Method." *Survey Methods for Nature Conservation* 2: 349–74.
- Gillison, Andrew, and Kenneth Brewer. 1985. "The Use of Gradient Directed Transects or Gradsects in Natural Resource Surveys." *Journal of Environmental Management* 20: 103–27.
- Glinka, Konstantin. 1927. "Dokuchaev's Ideas in the Development of Pedology and Cognate Sciences." *Russian Pedological Investigations* 1.
- Guisan, Antoine, and Niklaus Zimmermann. 2000. "Predictive Habitat Distribution Models in Ecology." *Ecological Modelling* 135 (2): 147–86. [https://doi.org/10.1016/S0304-3800\(00\)00354-9](https://doi.org/10.1016/S0304-3800(00)00354-9).
- Ham, Jisoo, Yangchi Chen, Melba Crawford, and Joydeep Ghosh. 2005. "Investigation of the Random Forest Framework for Classification of Hyperspectral Data." *IEEE Transactions on Geoscience and Remote Sensing* 43 (3): 492–501. <https://doi.org/10.1109/tgrs.2004.842481>.
- Heuvelink, G., and R. Webster. 2001. "Modelling Soil Variation: Past, Present, and Future." *Geoderma* 100 (3): 269–301. [https://doi.org/10.1016/S0016-7061\(01\)00025-8](https://doi.org/10.1016/S0016-7061(01)00025-8).
- Hijmans, Robert J. 2021. **terra**: *Spatial Data Analysis*. <https://CRAN.R-project.org/package=terra>.
- Hijmans, Robert J., Steven Phillips, John Leathwick, and Jane Elith. 2020. **dismo**: *Species Distribution Modeling*. <https://CRAN.R-project.org/package=dismo>.
- Hirzel, Alexandre, Véronique Helfer, and F. Metral. 2001. "Assessing Habitat-Suitability Models with a Virtual Species." *Ecological Modelling* 145 (2-3): 111–21. [https://doi.org/10.1016/S0304-3800\(01\)00396-9](https://doi.org/10.1016/S0304-3800(01)00396-9).
- Hudson, Berman. 1992. "The Soil Survey as Paradigm-Based Science." *Soil Science Society of America Journal* 56 (3): 836–41. <https://doi.org/10.2136/sssaj1992.03615995005600030027x>.
- Jasiewicz, Jaroslaw, Pawel Netzel, and Tomasz Stepinski. 2015. "GeoPAT: A Toolbox for Pattern-Based Information Retrieval from Large Geospatial Databases." *Computers & Geosciences* 80: 62–73. <https://doi.org/10.1016/j.cageo.2015.04.002>.
- Jasiewicz, Jaroslaw, Tomasz Stepinski, and Jacek Niesterowicz. 2018. "Multi-Scale Segmentation Algorithm for Pattern-Based Partitioning of Large Categorical Rasters." *Computers & Geosciences* 118: 122–30. <https://doi.org/10.1016/j.cageo.2018.06.003>.
- Kaufman, Leonard, and Peter Rousseeuw. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons.
- Knick, Steven, and Deanna Dyer. 1997. "Distribution of Black-Tailed Jackrabbit Habitat Determined by GIS in Southwestern Idaho." *The Journal of Wildlife Management*, 75–85. <https://doi.org/10.2307/3802416>.
- Kohonen, Teuvo. 1990. "The Self-Organizing Map." *Proceedings of the IEEE* 78 (9): 1464–80. <https://doi.org/10.1109/78.59163>.

- [//doi.org/10.1016/s0925-2312\(98\)00030-7](https://doi.org/10.1016/s0925-2312(98)00030-7).
- Leroy, Boris, Christine Meynard, Céline Bellard, and Franck Courchamp. 2016. "virtualspecies, an R Package to Generate Virtual Species Distributions." *Ecography* 39 (6): 599–607. <https://doi.org/10.1111/ecog.01388>.
- MacMillan, R., W. Pettapiece, S. Nolan, and T. Goddard. 2000. "A Generic Procedure for Automatically Segmenting Landforms into Landform Elements Using DEMs, Heuristic Rules and Fuzzy Logic." *Fuzzy Sets and Systems* 113 (1): 81–109. [https://doi.org/10.1016/S0165-0114\(99\)00014-7](https://doi.org/10.1016/S0165-0114(99)00014-7).
- Malczewski, Jacek. 2006. "GIS-Based Multicriteria Decision Analysis: A Survey of the Literature." *International Journal of Geographical Information Science* 20 (7): 703–26. <https://doi.org/10.1080/13658810600661508>.
- McBratney, Alex, M. L. Mendonça Santos, and Budiman Minasny. 2003. "On Digital Soil Mapping." *Geoderma* 117 (1-2): 3–52. [https://doi.org/10.1016/S0016-7061\(03\)00223-4](https://doi.org/10.1016/S0016-7061(03)00223-4).
- McSweeney, K., B. K. Slater, R. David Hammer, J. C. Bell, P. E. Gessler, and G. W. Petersen. 1994. "Towards a New Framework for Modeling the Soil-Landscape Continuum." *Factors of Soil Formation: A Fiftieth Anniversary Retrospective* 33: 127–45. <https://doi.org/10.2136/sssaspecpub33.c8>.
- Nolan, Connor, John Tipton, Robert K Booth, Mevin B Hooten, and Stephen T Jackson. 2019. "Comparing and Improving Methods for Reconstructing Peatland Water-Table Depth from Testate Amoebae." *The Holocene* 29 (8): 1350–61. <https://doi.org/10.1177/0959683619846969>.
- Nowosad, Jakub. 2021. "motif: An Open-Source R Tool for Pattern-Based Spatial Analysis." *Landscape Ecology* 36 (1): 29–43. <https://doi.org/10.1007/s10980-020-01135-0>.
- Nowosad, Jakub, and Tomasz Stepinski. 2021. "Pattern-Based Identification and Mapping of Landscape Types Using Multi-Thematic Data." *International Journal of Geographical Information Science* 35 (8): 1634–49. <https://doi.org/10.1080/13658816.2021.1893324>.
- Park, S. J., and N. Van De Giesen. 2004. "Soil-Landscape Delineation to Define Spatial Sampling Domains for Hillslope Hydrology." *Journal of Hydrology* 295 (1-4): 28–46. <https://doi.org/10.1016/j.jhydrol.2004.02.022>.
- Pebesma, Edzer. 2018. "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal* 10 (1): 439–46. <https://doi.org/10.32614/RJ-2018-009>.
- Pike, Richard. 1988. "The Geometric Signature: Quantifying Landslide-Terrain Types from Digital Elevation Models." *Mathematical Geology* 20 (5): 491–511. <https://doi.org/10.1007/BF00890333>.
- Pike, Richard, and Wesley Rozema. 1975. "Spectral Analysis of Landforms." *Annals of the Association of American Geographers* 65 (4): 499–516. <https://doi.org/10.1111/j.1467-8306.1975.tb01058.x>.
- Rodrigues, Marcos, Sergi Costafreda-Aumedes, Carles Comas, and Cristina Vega-García. 2019. "Spatial Stratification of Wildfire Drivers Towards Enhanced Definition of Large-Fire Regime Zoning and Fire Seasons." *Science of the Total Environment* 689: 634–44. <https://doi.org/10.1016/j.scitotenv.2019.06.467>.
- Schmidt, Jochen, Phil Tonkin, and Allan Hewitt. 2005. "Quantitative Soil - Landscape Models for the Haldon and Hurunui Soil Sets, New Zealand." *Soil Research* 43 (2): 127. <https://doi.org/10.1071/sr04074>.
- Scull, Peter, Janet Franklin, Oliver Chadwick, and D. McArthur. 2003. "Predictive Soil Mapping: A Review." *Progress in Physical Geography* 27 (2): 171–97. <https://doi.org/10.1191/0309133303pp366ra>.
- Tibshirani, Robert, Guenther Walther, and Trevor Hastie. 2001. "Estimating the Number of Clusters in a Data Set via the Gap Statistic." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63 (2): 411–23. <https://doi.org/10.1111/1467-9868.00293>.
- Watrous, Kristen, Therese Donovan, Ruth Mickey, Scott Darling, Alan Hicks, and Susanna Von Oettingen. 2006. "Predicting Minimum Habitat Characteristics for the Indiana Bat in the Champlain Valley." *The Journal of Wildlife Management* 70 (5): 1228–37. [https://doi.org/10.2193/0022-541X\(2006\)70%5B1228:PMHCFT%5D2.0.CO;2](https://doi.org/10.2193/0022-541X(2006)70%5B1228:PMHCFT%5D2.0.CO;2).
- Wehrens, Ron, and Johannes Krusselbrink. 2018. "Flexible Self-Organizing Maps in Kohonen 3.0." *Journal of Statistical Software* 87 (1): 1–18. <https://doi.org/10.18637/jss.v087.i07>.
- Wessels, K. J., S. Van Jaarsveld, J. D. Grimbeek, and M. J. Van der Linde. 1998. "An Evaluation of the Gradsect Biological Survey Method." *Biodiversity & Conservation* 7 (8): 1093–1121. <https://doi.org/10.1023/a:1008899802456>.
- West, Amanda, Sunil Kumar, Cynthia Brown, Thomas Stohlgren, and Jim Bromberg. 2016. "Field Validation of an Invasive Species Maxent Model." *Ecological Informatics* 36: 126–34. <https://doi.org/10.1016/j.ecoinf.2016.11.001>.
- Zadeh, Lofti A. 1965. "Fuzzy Sets." *Information and Control* 8 (1): 338–53. https://doi.org/10.1142/9789814261302_0001.
- Zhu, A. Xing. 1997. "A Similarity Model for Representing Soil Spatial Information." *Geoderma* 77 (2-4): 217–42. [https://doi.org/10.1016/S0016-7061\(97\)00023-2](https://doi.org/10.1016/S0016-7061(97)00023-2).
- Zhu, A. Xing, and Lawrence Band. 1994. "A Knowledge-Based Approach to Data Integration for Soil Mapping." *Canadian Journal of Remote Sensing* 20 (4): 408–18. <https://doi.org/10.1080/07038992.1994.10874583>.
- Zhu, A. Xing, Feng Qi, Amanda Moore, and James Burt. 2010. "Prediction of Soil Properties Using

Fuzzy Membership Values." *Geoderma* 158 (3-4): 199–206. <https://doi.org/10.1016/j.geoderma.2010.05.001>.

Zhu, A. Xing, Lin Yang, Baolin Li, Chengzhi Qin, Edward English, James Burt, and Chenghu Zhou. 2008. "Purposive Sampling for Digital Soil Mapping for Areas with Limited Data." In *Digital Soil Mapping with Limited Data*, 233–45. Springer. https://doi.org/10.1007/978-1-4020-8592-5_20.

Bryan A. Fuentes
University of Arkansas
Department of Crop, Soil, and Environmental Sciences
Fayetteville, Arkansas
ORCID: 0000-0003-3506-7101
bafuente@uark.edu

Minerva J. Dorantes
University of Arkansas
Department of Crop, Soil, and Environmental Sciences
Fayetteville, Arkansas
ORCID: 0000-0002-2877-832X
mjdorant@uark.edu

John R. Tipton
University of Arkansas
Department of Mathematical Sciences
Fayetteville, Arkansas
ORCID: 0000-0002-6135-8141
jrtipton@uark.edu

PDFEstimator: An R Package for Density Estimation and Analysis

by Jenny Farmer and Donald Jacobs

Abstract This article presents **PDFEstimator**, an R package for nonparametric probability density estimation and analysis, as both a practical enhancement and alternative to kernel-based estimators. **PDFEstimator** creates fast, highly accurate, data-driven probability density estimates for continuous random data through an intuitive interface. Excellent results are obtained for a diverse set of data distributions ranging from 10 to 10^6 samples when invoked with default parameter definitions in the absence of user directives. Additionally, the package contains methods for assessing the quality of any estimate, including robust plotting functions for detailed visualization and trouble-shooting. Usage of **PDFEstimator** is illustrated through a variety of examples, including comparisons to several kernel density methods.

1 Introduction

The ability to estimate a probability distribution from a single data sample is critical across diverse fields of science and finance (Munkhammar et al., 2017; Sidibé et al., 2018; Tang et al., 2016; Cavuoti et al., 2017). Estimating the parameters of the underlying density function becomes increasingly difficult when there is no prior information about the number of parameters, as the shape and complexity must also be inferred from that data. Although there are many nonparametric methods, kernel density estimation (KDE) is among the most popular.

Variants of KDE differ based on how they implement the selections of the bandwidth and the kernel function, which are nontrivial decisions that can have a significant impact on the quality and performance of the estimate. Most KDE implementations allow the user to manipulate some parameters manually, allowing for an experienced user to fine-tune the default behavior for improved results. Unfortunately, user directives introduce unavoidable subjectivity to the estimate. More advanced implementations include intelligent and adaptive bandwidth selection optimized according to the characteristics of the data (Chen, 2017; Botev et al., 2010). However, there is inevitably a trade-off between computational performance and accuracy (Gray and Moore, 2003). There are many R packages available that can estimate density nonparametrically, typically based on KDE (Calonico et al., 2019; Hayfield and Racine, 2008; Moss and Tveten, 2019; Konietschke et al., 2015; Sebastian et al., 2019; Wand and Ripley, 2021; Wand, 2021; Nagler and Vatter, 2020). The core R function `density` implements a straightforward kernel density method.

Presented in this article is the package **PDFEstimator** for nonparametric density estimation and analysis (Farmer and Jacobs, 2018). The features of **PDFEstimator** can be separated into two categories. The first is a novel estimation method based on the principle of maximum entropy, available through the `estimatePDF` function. A primary advantage of `estimatePDF` is the automated interface, requiring nothing from the user other than a data sample. Range, multi-scale resolution, outliers, and boundaries are determined within the algorithm to achieve optimized data-driven estimates appropriate to the given sample. Although these defaults can be overridden by a sophisticated user, overrides generally do not improve the results. Additionally, multiple acceptable solutions can be returned, which is particularly useful in the case of low sample sizes where there is more statistical uncertainty.

The second category of features included in **PDFEstimator** is a unique set of assessment utilities for evaluating the accuracy of the solution and highlighting areas of uncertainty within a density estimate. Furthermore, a user-defined threshold can be specified to identify data points that fall outside of an expected confidence level. These diagnostics tools are visualized through a variety of customized plotting options, thus aiding in the evaluation of difficult distributions. Most importantly, all of these features can be applied towards any estimation function, such as `density`, as they are universal measurements independent of the method used, allowing for an integrated comparison between alternative models.

The remainder of this paper is organized as follows. [Available functions and usage](#) describes the functions available in this package, including their usage and underlying methods. [Examples and illustrations](#) provides additional and advanced examples for identifying and troubleshooting problems with any density estimation method. [Comparison to kernel-based estimators](#) compares `estimatePDF` with three popular kernel-based estimation packages using a diverse set of known distributions. Finally, [The 1872 Hidalgo stamp issue of Mexico](#) demonstrates the use of **PDFEstimator** for a well-known real data set from a rare stamp collection.

Function	Description
getTarget	Returns upper and lower limits of SQR for a given target level
plotBeta	Plots a shaded region outlining expected range of SQR values by position
estimatePDF	Estimates a density from a data sample. Returns a PDFe estimation object.
convertToPDFe	Converts any PDF to a PDFe estimation object for diagnostic purposes.
approximatePoints	Returns approximated PDF for an existing PDFe estimation object at a given set of data points.
plot	Main plotting function for PDFe estimation objects.
lines	Plots the density for a PDFe estimation object as a connecting line segment to an existing plot.
summary	Prints a summary of the PDFe object.
print	Prints the probability density and cumulative density for each estimation point in the PDFe object.

Table 1: Overview of functions in the **PDFEstimator** package.

2 Available functions and usage

An overview of the functions included in **PDFEstimator** is listed in Table 1. A critical component of the tools in **PDFEstimator** is a PDFe object, which encapsulates all information necessary for plotting and assessing the quality of any density estimate. The member variables for the PDFe class definition are listed in Table 2. The methods for calculating the necessary components of the PDFe and how they are employed in each of the functions in Table 1 are described in this section.

The PDFe class

The first four member variables listed in Table 2 are commonly understood values for an estimated density, beginning with the random data sample that is the basis for the estimate. The probability density function (PDF) is estimated for the range of points defined in x . Similarly, the cumulative distribution function (CDF) can be calculated representing the cumulative probability of the PDF for each x . **PDFEstimator** defines additional descriptions for an estimate that measure the quality of its fit to the sample data. Central to the quality of these estimates is a scoring mechanism to rate the overall fit of an estimate to the sample. A single average score is calculated, as well as individual confidence levels for each data point within the sample. These scores are based on order statistics (Wilks, 1948).

If the CDF, defined on the range (0 1), is an accurate representation of the data, then $CDF(x)$ will represent uniform random data. The general problem then becomes assessing if $r_k = CDF(x_k)$ represents uniform data for a given sample. Although there are many methods to test for uniform data (Farmer et al., 2019), **PDFEstimator** employs an average quadratic z-score, defined as

$$z^2 = \frac{-1}{N} \sum_{k=1}^N \frac{(r_k - \mu_k)^2}{\sigma_k^2}, \quad (1)$$

where k is the sort ordered position in a sample size N , and μ_k and σ_k are the mean and standard deviation from single order statistics known to be $\mu_k = -\frac{k}{N+1}$ and $\sigma_k = \frac{\mu_k(\mu_k-1)}{\sqrt{N+2}}$. Perfectly uniform data would yield a score of exactly zero.

To study typical z-scores for uniform random data, extensive numerical experiments were generated with a random number generator on the range (0, 1) for many different sample sizes. The typical distribution of scores according to Equation 1 is represented in the left plot for Figure 1. The peak density of the PDF corresponds to the most likely z-score and occurs at a value of approximately -0.5, with a sharp drop-off in the density as scores approach zero. The threshold value of the PDFe object reports the empirical cumulative probability for the z-score, as a percentage, shown on the right-hand plot of Figure 1. The cumulative probability for the peak z-score is calculated to be a little over 0.7, thus a threshold value near or greater than 70% can be considered a highly probable fit. A threshold of less than 5%, by contrast, is low probability and therefore likely an underfit for the data. In this event, the PDFe member variable `failedSolution` is set to TRUE. A score with a threshold of 95%, however, is similarly unlikely and can be interpreted as overfitting the data. The software does not rigidly enforce

Member Variable	Description
sample	Sample data used to estimate the density.
x	Points where the density is estimated.
pdf	Estimated density values for x.
cdf	Cumulative density values for x.
threshold	Threshold score, expressed as a percentage, measuring the uniformity of the CDF of sample data.
failedSolution	If true, indicates that the estimate returned does not meet at least a 5% threshold.
sqr	Scaled quantile residual values for sample data points.
sqrSize	Size of sqr.

Table 2: Member variables of the PDFe class.

a particular score upon an accepted solution, but rather uses the numerically calculated density shown in Figure 1 as a guide to iteratively move towards increasingly probable solutions.

The threshold provides an average score for the estimate, but to assess the estimate per position and identify the locations of potential errors, a scaled quantile residual (SQR) is defined as

$$SQR_k = \sqrt{N+2} (r_k - \mu_k). \quad (2)$$

The scaling factor of $\sqrt{N+2}$ creates a sample-size-invariant metric for each position k . The `sqr` member variable of the PDFe class contains a vector of SQR values according to Equation 2 and their ability to diagnose problems in a density estimate will be demonstrated with the `plot` function. Note that the PDFe object will also report the size of `sqr`, which will be the number of samples less the number of any outliers detected.

getTarget and plotBeta

It has been shown that, when plotted against position, SQR_k for uniform random data falls approximately within an oval shaped region (Farmer et al., 2019). The reason for this can be understood by examining the beta distributions that govern order statistics for sort ordered random uniform data (Wilks, 1948). The probability of u for position k in uniform random data with N samples is as follows.

$$p_k(u) = \frac{N!}{(k-1)!(N-k)!} u^{k-1} (1-u)^{N-k} \quad (3)$$

By integrating Equation 3 for each position k , confidence levels for SQR values of sample size N are calculated in the `getTarget` function. Figure 2 demonstrates confidence levels for three different target percentages, plotted as contour lines. The background shading in grey represents typical ranges of SQR values, with darker shading corresponding to higher probability areas. This shading is independent of sample size, and is calculated according to Equations 2 and 3 in the `plotBeta` function.

estimatePDF

`estimatePDF` provides an R interface for nonparametric density estimation based on a novel method providing an alternative to traditional KDE implementations. Details of this approach, based on the principle of maximum entropy (Wu, 1997), were published previously and have been shown to produce more accurate estimates than KDE in most cases (Farmer and Jacobs, 2018; Farmer et al., 2019; Puchert et al., 2021; Farmer and Jacobs, 2022). For optimal performance and flexibility with other applications, this functionality is performed within a set of C++ classes and is not the focus of this paper, but a brief summary is provided for insight into the `estimatePDF` interface.

In a traditional maximum entropy method, moments for a set of characteristic functions are introduced, and the coefficients to these functions are optimized to match the predicted moments with the empirical moments. As a particular choice of moments that exist for any probability density, and to form a systematic truncated expansion over a complete set of orthogonal functions, the analytical form for the density function from a data sample is expressed as

$$p(v) = \sum_{j=1}^D \exp(\lambda_j g_j(v)), \quad (4)$$

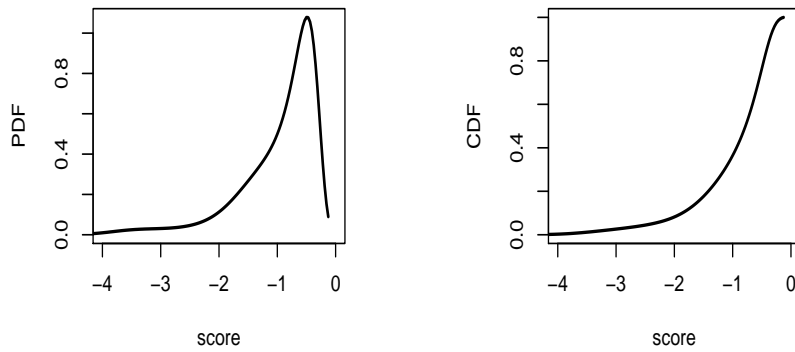


Figure 1: These plots demonstrate the distribution of scores as defined by Equation 1 and are used to assess the probability of a trial solution. The density was estimated empirically by generating 1000 trials of 10,000 uniform random data samples. The probability density function is shown on the left and the cumulative density function is on the right.

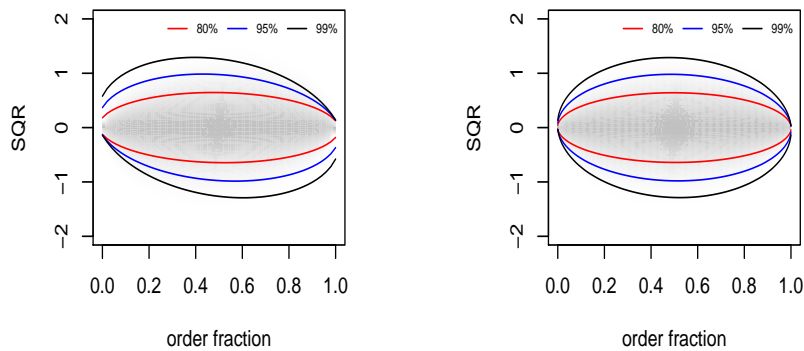


Figure 2: Confidence levels by position for a scaled quantile residual plot based on beta distribution probabilities according to single order statistics for a sample size of 50 (left) and 1000 (right). The colored contour lines define regions within an oval shape that give target levels for observing data points deviating away from perfect uniform spacing as expected for true uniform random data. Note that the skewness in the contour lines are noticeable for small sample sizes compared to large sample sizes.

where $g_j(v)$ are bounded level functions and λ_j are Lagrange coefficients controlling the shape of the density function (Jacobs, 2008). For a fixed number of coefficients, D , this method is parametric in form. Although solving for the coefficients analytically is increasingly impractical for high dimensionality, a random search method is employed that provides very efficient numerical optimization. An expansion of orthogonal functions is constructed in the form of Equation 4, without specifying D in advance, where higher mode orthogonal functions are successively added as needed. The algorithm iteratively explores possible density functions by perturbing the Lagrange coefficients that are currently present, while D is slowly increased, testing each possibility according to the scoring function in Equation 1 to converge towards an accurate estimate.

The arguments for estimatePDF are listed in Table 3. If no other parameters are specified, the range of the returned estimate is calculated automatically. By default, left and right boundaries are presumed theoretically infinite and allowed to extend beyond the range of the data sample. The finite numerical bounds are calculated according to the density near the tails, with longer tails receiving more padding than shorter tails. Similarly, extreme outliers are detected and removed from the sample

Arguments	Description
sample	A vector containing the data sample to estimate.
pdfLength	Specifies the desired length of the estimate returned. By default, this length is calculated based on the length of the sample.
estimationPoints	An optional vector containing specific points to estimate.
lowerBound	Sets the finite lower bound for the sample, if it exists.
upperBound	Sets the finite upper bound for the sample, if it exists.
lagrangeMin	Specifies the minimum allowed dimension, D , in Equation 4.
lagrangeMax	Specifies the maximum allowed dimension, D , in Equation 4.
debug	If TRUE, detailed progress will be printed to the console.
outlierCutoff	If greater than 0, specifies the range of included sample data, according to the formula $[(Q1 - outlierCutoff \times IQR), (Q3 + outlierCutoff \times IQR)]$, where $Q1$, $Q3$, and IQR represent the first quartile, third quartile, and inter-quartile range, respectively.
target	Sets a target percentage threshold between 0 and 100. The default is 70, the minimum accepted is 5.
smooth	If TRUE (default), preference is given towards smooth density estimates.

Table 3: Overview of arguments in the estimatePDF function.

as appropriate according to the outlierCutoff argument. Alternatively, the lowerBound, upperBound, and outlierCutoff parameters can be independently specified to provide the user complete control over the range of the estimate. Setting outlierCutoff to zero turns off outlier detection and includes all the data.

The number of expansions in Equation 4 begins at lagrangeMin and is capped at lagrangeMax, with default values of 1 and 200, respectively. The maximum of 200 provides a generous realistic upper limit to the complexity of the estimate and the computational time required but can be altered to either increase accuracy or decrease compute time. Another reason to override these limits is to create a semi-parametric estimate by narrowing the range between minimum and maximum. A strictly parametric approach can be achieved by setting the two limits to the same value. For example, setting both lagrangeMin and lagrangeMax to 1 forces a uniform fit. Similarly, setting them both to either 2 or 3 respectively yields exponential and Gaussian distributions.

The target argument refers to the cumulative probability of the z-score, as previously discussed. Note that target is a user-defined argument, whereas threshold in the PDFe class is the actual threshold achieved. These values may be different for several reasons. For example, the search for the target threshold may abort prematurely if the lagrangeMax has been exceeded or if progress has been stalled. Additionally, a small penalty is added to the score if the estimate becomes exceptionally noisy in areas of low density where sharp features are not justified. Therefore, a smoother curve may be favored over a higher score. The smoothing penalty is constructed according to a Taylor expansion error estimate as described previously (Jacobs, 2008). The original model calculated a second order expansion, but a first order approximation was found to be sufficient and implemented in this version. This behavior can be circumvented when intentionally searching for small peaks by setting the smooth argument to FALSE.

convertToPDFe and approximatePoints

The estimatePDF function performs the density estimation based on the input parameters in Table 3 and returns a PDFe object for plotting and additional analysis. Alternatively, the convertToPDFe function will create a PDFe object for an estimate calculated using any other method for a given data sample. convertToPDFe requires a data sample and the (x, y) values for the estimate, and calculates the score, threshold, and SQR values for each point in the sample.

The approximatePoints function operates on a PDFe object to approximate the PDF for additional data points after the estimate has already been calculated using either estimatePDF or convertToPDFe. This functionality is similar to specifying estimationPoints in the estimatePDF function, but is provided for the convenience of approximating different points without having to recalculate the estimate. The following example will create a PDFe object for a KDE estimate of a random sample from a standard normal distribution, and then return additional density approximations at points -3, 0, and 1.

```

sample = rnorm(1000)
kde = density(sample)
pdfe = convertToPDFe(sample, kde$x, kde$y)
approximatePoints(pdfe, c(-3, 0, 1))

```

plot

The `PDFEstimator::plot.PDFe` function extends the generic `plot` function in R supporting all existing graphical parameters, with additional options summarized in Table 4. The first argument listed in the table is the `PDFe` object returned by `estimatePDF` or `convertToPDFe` and is required for all plots. The `plotPDF` and `plotSQR` arguments can be independently set to `TRUE` or `FALSE` and collectively control the plot type. The `plotShading` and `showOutlierPercent` values invoke the `plotBeta` and `getTarget` functions from Table 1 and provide optional diagnostics to highlight specified uncertainties within the estimate through the use of the SQR plot. The remaining arguments listed in Table 4 control minor graphical features for customized aesthetics. The following examples will demonstrate the variety of plots that can be created with combinations of these options.

Arguments	Description
<code>x</code>	A <code>PDFe</code> estimation object. Returned from <code>estimatePDF</code> .
<code>plotPDF</code>	Plots the probability density for <code>x</code> if <code>TRUE</code> .
<code>plotSQR</code>	Plots the scaled quantile residual (SQR) for <code>x</code> if <code>TRUE</code> . If <code>plotPDF</code> is also <code>TRUE</code> , the SQR will be scaled to the range of the density.
<code>plotShading</code>	Plots gray background shading indicating approximate confidence levels for the SQR, where darker shades indicate higher confidence. Setting this to <code>TRUE</code> only has meaning when <code>plotSQR = TRUE</code> .
<code>shadeResolution</code>	Specifies the number of data points plotted in the background shading when <code>plotShading = TRUE</code> . Increasing this resolution will create sharper and more accurate approximations for the confidence levels, but will take more time to plot.
<code>showOutlierPercent</code>	Specifies the threshold to define outliers for SQR. Must be a number between 1 and 100.
<code>outlierColor</code>	Specifies the color for outliers when <code>showOutlierPercent</code> is defined.
<code>sqrPlotThreshold</code>	Magnitude of y-axis for SQR plot.
<code>sqrColor</code>	Specifies the SQR color for non-outliers
<code>type</code>	Specifies the line type of the density curve if <code>plotPDF = TRUE</code> . If <code>plotPDF = FALSE</code> and <code>plotSQR = TRUE</code> , the SQR plot uses this type. The default is <code>lines</code> type.
<code>lwd</code>	Specifies the line width of the density curve if <code>plotPDF = TRUE</code> . If <code>plotPDF = FALSE</code> and <code>plotSQR = true</code> , the SQR plot uses this width. The default is 2.
<code>xlab</code>	x-axis label for pdf. If <code>plotPDF = FALSE</code> and <code>plotSQR = TRUE</code> , then the <code>sqr</code> plot uses this label.
<code>ylab</code>	y-axis label for pdf. If <code>plotPDF = FALSE</code> and <code>plotSQR = TRUE</code> , then the <code>sqr</code> plot uses this label.
<code>legendcex</code>	expansion factor for legend point size with <code>sqr</code> plot type, for <code>plotPDF = FALSE</code> and <code>plotSQR = TRUE</code> .
...	Inherits all other plotting arguments from <code>plot</code> function

Table 4: Overview of arguments in the `plot` function.

3 Examples and illustrations

Plotting estimates

Figure 3 contains two separate examples of the Maxwell distribution. The left panel demonstrates the `plot` function using all the default parameters and simply plots the density estimate. On the right, the `plotSQR` parameter creates the SQR plot, as described in Equation 2, overlaying the density. A shaded background, scaled according to the data sample, is plotted with the addition of the `plotShading`

parameter set to TRUE, providing a visual approximation of the most probable range of scaled quantile residual values. Finally, the `showOutlierPercent` parameter flags scaled quantile residual values that are outside of the 99% target interval. By default, these outliers are plotted in red.

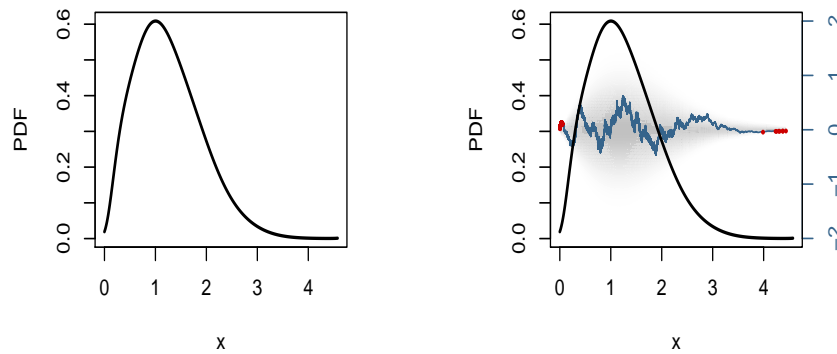


Figure 3: (left) Density estimates for the Maxwell distribution with 100,000 samples using default parameters. (right) In addition to the density estimate, the scaled quantile residual mapped to the original variable x is shown along with a shaded region indicating 99% target. The scaled quantile residuals outside of the 99% target interval are highlighted in red.

Figure 4 contains additional examples for visual assessment of estimates. Each of these plots is based on the sawtooth distribution, defined by ten identical isosceles triangles at equal intervals. The sawtooth distribution is designed to be challenging to estimate due to these extremely sharp peaks. The exact distribution is plotted in gray in the top left panel of Figure 4, with the `estimatePDF` estimate for 100,000 samples shown in black. The estimate captures the high peaks well but falls short of reaching the lowest points of each triangle. The top right plot demonstrates how the `showOutlierPercent` parameter can identify specific areas of lower confidence in the estimate when the exact distribution is not known. In this example, the estimate is plotted in gray with green highlighting the sample points outside of an 80% target level. Although the estimate closely approximates the distribution, the low peaks are identified as less accurate.

The bottom row of figure 4 shows an alternative diagnostic visualization of the same estimate. These plots do not include the density estimate, but show only the SQR plot. Note that when the SQR is plotted alone, each sample point is spaced equally according to sort ordered position. Additionally, dotted lines represent the confidence interval, if `showOutlierPercent` is specified, and the calculated percentage of points lying outside this threshold are printed on the bottom right. These examples show targeted thresholds of 80% and 99% with outliers representing 8% and 1% of the points, respectively. If the sum of the threshold and outlier percentages fall far below or above 100%, this indicates a poor estimate. The figures in this section collectively demonstrate a wide range of visual assessment and color choices available with this customized plot function.

Advanced diagnostics

The results returned from `estimatePDF` provide additional information for further analysis when needed. For example, the first plot in Figure 5 is the SQR result for 10,000 data samples generated from a normal-cubed distribution. In this case, the `failedSolution` return value is TRUE, indicating that the estimate does not meet the required threshold for the scoring function. Rather than return a NULL solution in the event of an unacceptable score, the best scoring estimate is always returned. However, in addition to the poor average score, the SQR plot also shows large variations outside of a 99% threshold.

The second plot in Figure 5 shows the SQR result for the same sample data estimated with the core R function density. The density estimate is first converted to a `PDFe` object via the `convertToPDFe` function. The `failedSolution` return variable indicates this estimate also does not meet an acceptable fit, but the SQR plot further suggests an extreme underfit of the estimate near the midpoint of the sample. This is a common weakness of KDE estimates for data with sharp peaks and long tails. In fact, the normal-cubed distribution is undefined at zero and diverges as it approaches this singularity

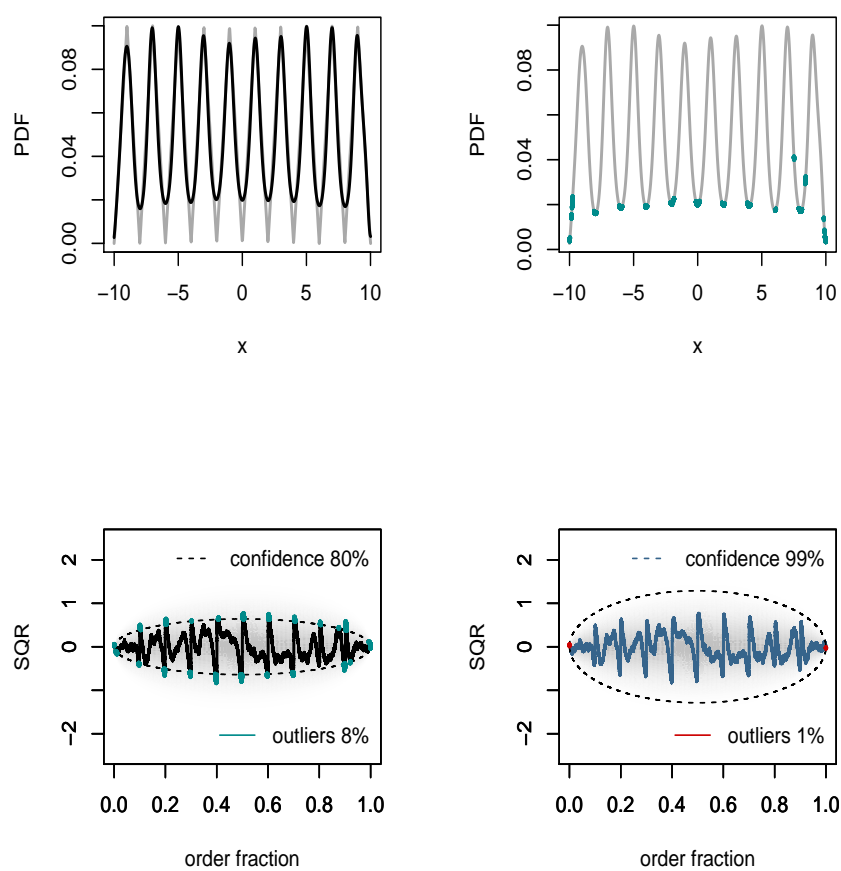


Figure 4: Density estimates for the sawtooth distribution with 100,000 samples. The top left plot shows the exact known distribution in gray and the estimate in black. The top right plot shows the estimate in gray, with green highlights indicating areas outside of the 80% target threshold. The bottom row demonstrates scaled quantile residual plots for the same estimate, for 80% and 99% target levels.

from both directions, presenting a challenge for any density estimator. In either method, however, the information available in PDFe alerts the user of problems that may require intervention.

A manual inspection of the sample data, perhaps in the form of a quantile or histogram plot, confirms an approximately symmetric distribution of data with an extreme variation in density near the center. A reasonable course of action in this event is to attempt to fit the low-density tails separately from the high-density peak. Figure 6 demonstrates this approach, modeling the distribution with three separate calls to `estimatePDF` (top) and `density` (bottom) for ranges delineated by -0.01 and 0.01 around the peak. The tails are bounded according to the range returned from the respective original estimates for each method.

For `estimatePDF`, this produces individual estimates mostly within the 99% target range shown in the three SQR plots in Figure 6. This example suggests a general divide and conquer method for addressing extreme distributions that is part of an automated procedure (to be published elsewhere). For `density`, however, the estimates remain poor in comparison even when fitting the regions separately. This is due, in part, because KDE methods do not incorporate automatic outlier detection. Additionally, KDE tends to perform poorly at boundaries, causing them to be less amenable to piecing together estimates in this way.

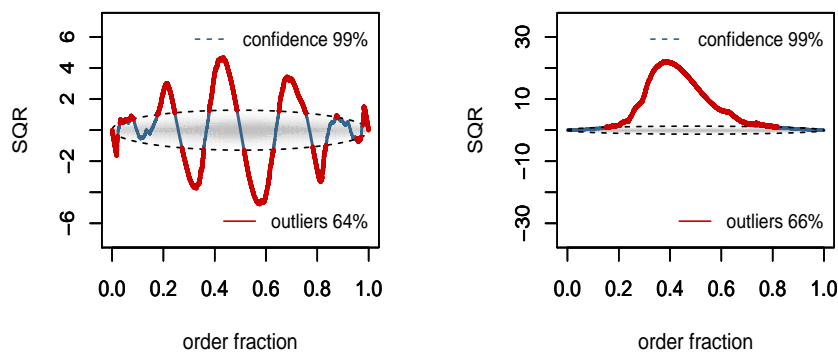


Figure 5: Scaled quantile residual plots for estimates of the normal-cubed distribution with 10,000 samples using `estimatePDF` (left) and `density` (right).

4 Comparison to kernel-based estimators

Conducting a fair comparison between nonparametric methods can be challenging due to the unrestrained nature of the input data. It is inevitable that different methods will do well estimating certain types of distributions while performing poorly on others. The `benchden` R package was implemented specifically to address this concern and facilitate an unbiased comparison between nonparametric methods (Mildenberger and Weinert, 2012). `Benchden` includes detailed information about a collection of 28 diverse known distributions, deliberately chosen to challenge estimation methods in a variety of ways, including long tails, discontinuities, and sharp or infinite peaks. This data set is used to collect statistics on the accuracy, computational performance, and usability of `estimatePDF` compared to several kernel-based methods.

Usability is admittedly somewhat subjective and difficult to measure. An advanced user with in-depth knowledge of the data and experience with a particular method may choose to set parameters that differ from default values. To control for expert user biases, default settings are used across all methods. Comparison functions were selected, in part, with the criteria that the only required user input is a data sample. However, all methods included in this comparison also allow optional parameters to specify finite boundary support for a distribution when provided by the `benchden` package. Although many kernel-based functions were evaluated and tested, three representative methods were selected for the results in this section. The first is `density`, the core R function. The other two are `npudens` and `bkde`, from the `np` (Hayfield and Racine, 2008) and `KernSmooth` (Wand and Ripley, 2021) packages, respectively. These packages were also chosen to demonstrate specific advantages and features.

Computational performance and accuracy comparisons are relatively straightforward to measure.

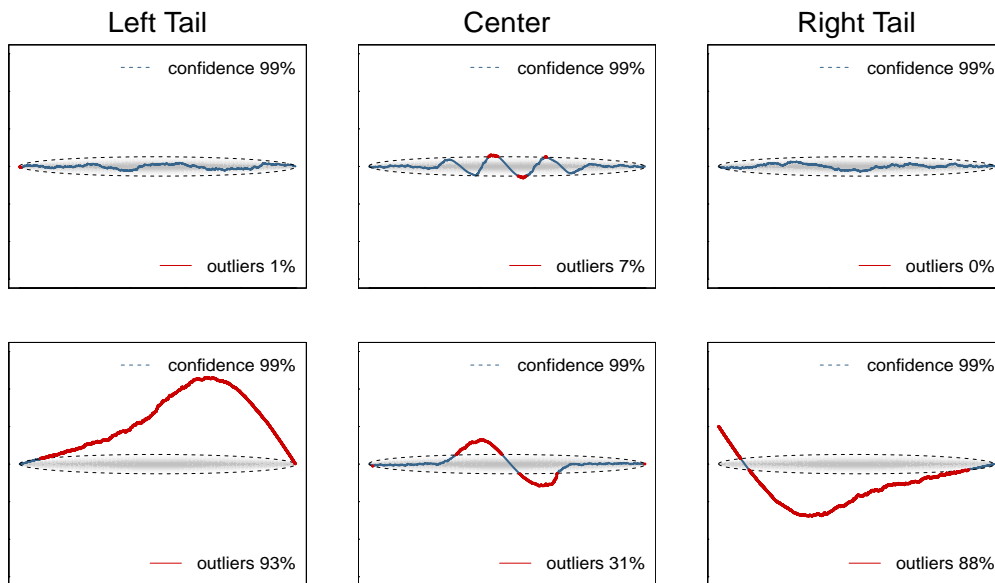


Figure 6: Scaled quantile residual plots for estimates of the normal-cubed distribution with 10,000 samples separated by left tail, middle, and right tail for estimatePDF (top row) and density (bottom row)

The R package **philentropy** (Drost, 2018) implements 46 measurements designed specifically for comparing two distributions. **Philentropy** was used together with **benchden** to generate random samples for each distribution and compare estimates for these samples against the known density. For each of the four selected nonparametric methods, 100 random samples from each distribution available in **benchden** were estimated, timed, and averaged for sample sizes ranging from 10 to 10^6 for all 46 accuracy measures in **philentropy**. All measurements were averaged first over 100 randomizations and then over all 28 distributions. Comparative results were qualitatively consistent between the 46 measures, particularly when averaged over a range of distributions, therefore, for simplicity, a single measure was selected for the plots in this section. A symmetric version of the chi-squared family was selected for the divergence, defined as

$$d = \sum_{i=1}^n \frac{(P_i - Q_i)^2}{P_i + Q_i}. \tag{5}$$

Representative results for ten trials of each of the 28 distributions for select sample sizes are shown in the left plot of Figure 7.

For functions `density`, `bkde`, and `estimatePDF`, the chi-squared error decreases with sample size while computational time, shown in the right plot of Figure 7, increases, as expected. The relative comparison between these three methods shows that `estimatePDF`, on average, produces more accurate estimates at the expense of increased computational time. These trends generally agree with more comprehensive comparisons between `estimatePDF` and other nonparametric methods (Puchert et al., 2021). `density` and `bkde` are very similar to one another, with `density` marginally slower and more accurate on average over `bkde`. The performance of `npudens` is less consistent. The **np** package allows for an adaptive bandwidth selection that is the default method for `npudens`. Although this functionality produces results with greater accuracy than `density` and `bkde`, it becomes computationally intractable as sample size increases. The authors recommend against using the adaptive bandwidth option for sample sizes beyond 1000. Simulations (not shown) were pushed to 50,000 samples at 100 trials, and 500,000 samples for a single trial, confirming the prediction of $O(n^2)$ time complexity for this method (Hayfield and Racine, 2008).

Although the plots in Figure 7 provide an excellent snapshot of overall trends for comparison, a more detailed analysis per distribution is necessary for practical insight into the strengths and weaknesses of `estimatePDF` compared to the kernel-based estimates. When viewing accuracy for each distribution separately, 12 of the 28 showed no significant differences between methods. When averaged over many trials, the magnitude of error empirically converges toward zero by approximately $n^{-0.56}$. The average chi-squared measures as a function of sample size for these 12 distributions are shown in the top left plot in Figure 8. Simple, well-behaved distributions are easy to estimate for all methods considered in this work.

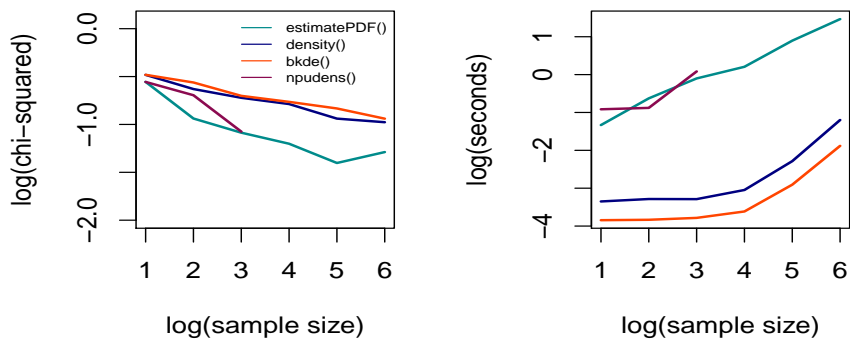


Figure 7: Comparison between four nonparametric estimators as a function of sample size according to accuracy (left) and performance (right). All quantities are averaged over 28 distributions for sample sizes in powers of 10.

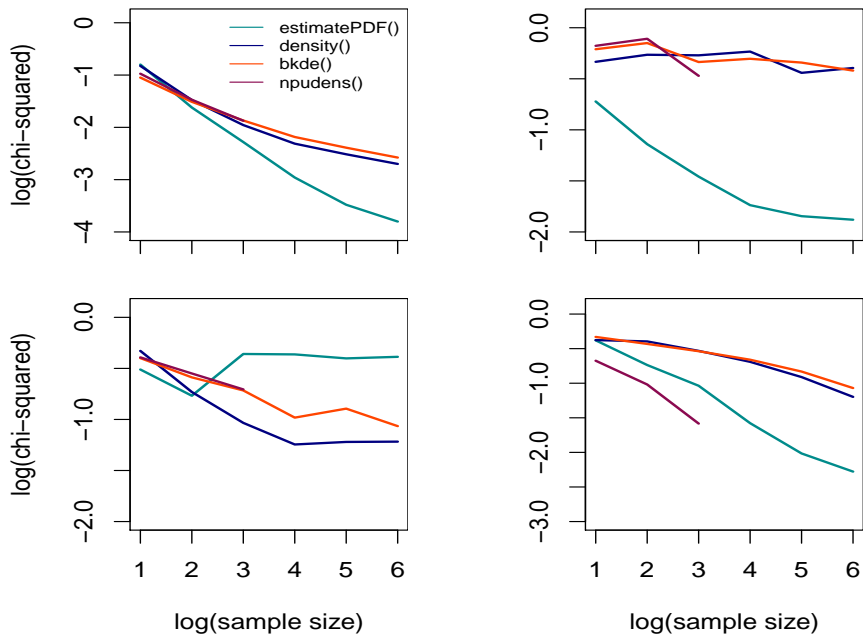


Figure 8: Accuracy comparison between four nonparametric estimators as a function of sample size averaged over subsets of distributions. Top left: distributions with high accuracy; top right: long tailed distributions; bottom left: Matterhorn and normal cubed distributions; bottom right: distributions with multiple peaks.

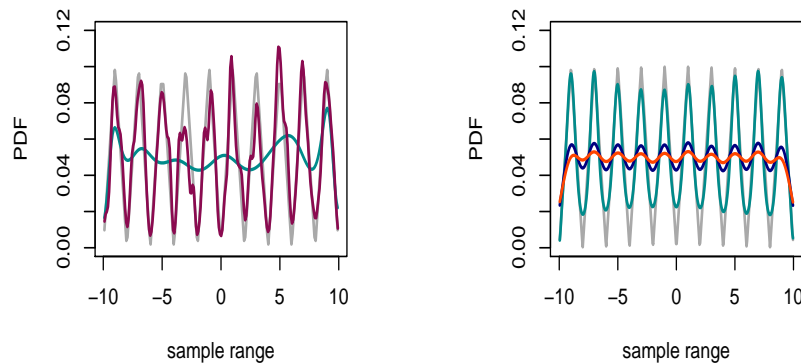


Figure 9: Left: Density estimates for the sawtooth distribution with 1000 samples for `estimatePDF` (green) and `npudens` (red). Right: Density estimates for the sawtooth distribution with 1 million samples for `estimatePDF` (green), `density` (blue), and `bkde` (orange).

The remaining plots in Figure 8 illustrate more interesting cases where `estimatePDF` and kernel-based methods show their differences. The top right plot, for example, is the accuracy averaged over four distributions (Cauchy, Pareto, symmetric Pareto, and inverse exponential) with extremely long tails. Error remains high and somewhat erratic for all KDE methods, and visual inspection of the density plots reveal that they will often miss the location and density of the peak in favor of attempting to capture the low density in the tails. The automatic boundary and outlier detection in the default behavior of `estimatePDF` correctly identifies the tails with near-zero density and removes them from consideration, thus fitting the peak extremely well with very little overall error in the estimate.

The bottom left plot in Figure 8, by contrast, shows the average accuracy of the two distributions (Matterhorn and normal-cubed) where `estimatePDF` performs poorly compared to the KDE methods. Although these distributions have quite different definitions, the common characteristics are that they are symmetrically distributed and are neither in L_2 nor L_∞ , with infinite peaks approaching zero from both directions. This particular set of features is unique among the 28 distributions in `benchden`, and poses an exceptional challenge to `estimatePDF`. The Matterhorn distribution, by far the worst performer of the two, additionally suffers from known machine-precision errors in the random number generator in `benchden`, occasionally producing samples equal to zero where the distribution is undefined (Mildenberger and Weinert, 2012). Illustrative results from `estimatePDF` for the normal-cubed distribution were previously shown in Figure 5, along with a demonstration of how a knowledgeable user can fit together segments of the data and combine the solutions to obtain a good estimate.

The bottom right plot in Figure 8 highlights four distributions (sawtooth, smooth comb, Marronite, and claw) with multiple sharp peaks. The adaptive bandwidth for `npudens` provides a clear advantage in accurately estimating these distributions over other KDE functions. For small sample sizes, `npudens` also maintains an advantage over `estimatePDF`. An example of this advantage is shown in the left panel of Figure 9, comparing `npudens` and `estimatePDF` for 1000 samples of the sawtooth distribution. The right panel for Figure 9 shows the estimates for this distribution at one million samples for `estimatePDF`, `density`, and `bkde`. The two kernel-based estimates, processing in under 1 second, are quite poor. The estimate for `estimatePDF`, taking about 1 minute, is notably improved and likely would be considered worth the additional computational investment to a user. The `npudens` estimate, however, is projected to require 10-12 days to compute for 1 million samples, assuming the continuation of $O(n^2)$ time increase. Any marginal increase in accuracy is unlikely to be worth the wait.

5 The 1872 Hidalgo stamp issue of Mexico

In 1988, Izenman and Sommer published a detailed statistical study and historical account describing a postage stamp collection issued in Mexico in 1872 (Izenman and Sommer, 1988). This particular collection consists of only 485 stamps preserved from the millions originally issued that year, providing a very small and rare sampling of the data. At this time in history, stamps were printed on a variety of paper types, with poor documentation and quality control on the thicknesses used for printing.

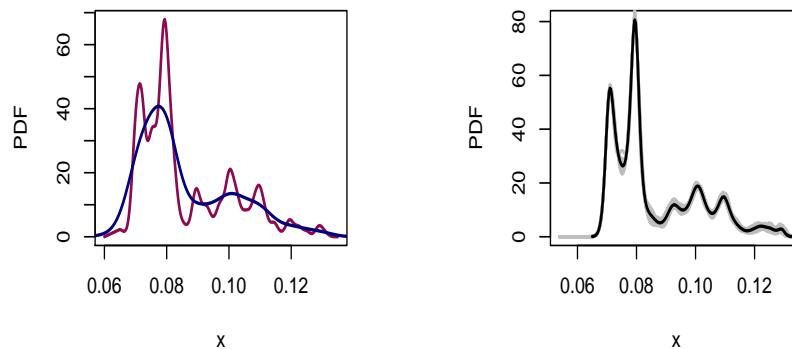


Figure 10: Estimates for the 1872 Hidalgo stamp issue of Mexico. (left) Default KDE estimate for density in blue, variable KDE estimate for `npudens` in red. (right) Multiple estimates for `estimatePDF` shown in gray, average in black.

Therefore, it is unknown how many paper types were used for this stamp issue in 1872, but those citing historical evidence and statistical analysis have estimated anywhere from 3 to 8 plausibly distinct thicknesses (Basford et al., 1997; Brewer, 2000; Izenman and Sommer, 1988; Sheather, 1992).

A straightforward default estimate using the density function, shown in blue on the left plot of Figure 10, yields only two modes, with a bandwidth of 0.00391. Converting this estimate to a `PDFe` object calculates a threshold of only 0.27%, indicating a very poor fit. By contrast, the `npudens` variable bandwidth estimate, shown in red, calculates a much smaller bandwidth at 0.00104 and includes at least 10 major and minor modes. Converting this estimate to a `PDFe` object yields a much more probable threshold of 56.4%. More sophisticated parametric methods, such as mixed normal mode analysis, suggest that 7 modes is optimal (Basford et al., 1997; Izenman and Sommer, 1988).

The inherent difficulty in estimating real world data from such a small sample size is in distinguishing true characteristics of the data from random fluctuations of the sample. A unique advantage of `estimatePDF` is the ability to produce multiple viable solutions to fit the sample data. Since `estimatePDF` employs a random process for optimizing its parameters, starting with different seeds can result in a range of unique possible solutions consistent with the sampled data. When sample sizes become large, multiple solutions generally will converge to one single solution, but estimates for small sample sizes will generally have more variation. This application, where small features are of critical interest, provides an example of when the user may wish to deactivate the smooth functionality. Removing this penalty from the score will prioritize the most likely fit to the data with no regard to spurious noise in low-density areas.

This effect is demonstrated in the right plot of Figure 10. The results of 20 calls to `estimatePDF` are plotted in gray, and the average density over these 20 estimates is plotted in black. The average shows 7 smooth, distinct modes, corresponding to the 7 paper types proposed by Izenman and Sommer based on historical evidence. However, even with smoothing disabled, the two peaks in the right tail are very small and account for some amount of variation from one estimate to the next. There is some support, both historical and statistical, that these two modes are not justifiable, with some argument that there are only 5 modes (Brewer, 2000; Sheather, 1992). By contrast, the small mode between the largest two peaks on the left, seen in the `npudens` estimate and in some of the `estimatePDF` estimates, is generally agreed to be a random fluctuation of the sample set. `estimatePDF` demonstrates that these fluctuations are all possible fits to the data.

6 Summary

`PDFestimator` is a probability density estimation package that introduces an R implementation of a novel nonparametric method, `estimatePDF`, based on maximum entropy. Any computational method must strike a reasonable balance between usability, computational time, and practical functionality. For comparison, `estimatePDF` was tested across a large range of random samples for 28 known distributions and compared to other popular R packages that implement nonparametric estimation through kernel density methods. Although specific results are problem-dependent, `estimatePDF`

generally computes estimates much more quickly than those with competitive accuracy. Included in **PDFEstimator** is a collection of scoring assessments and plotting functions for displaying results and identifying problem areas in the estimate. These advanced plotting and analysis functions are independent of distribution type and estimation method and can be applied towards any density estimator in R.

Bibliography

- K. E. Basford, G. J. McLachlan, and M. G. York. Modelling the distribution of stamp paper thickness via finite normal mixtures: The 1872 hidalgo stamp issue of mexico revisited. *Journal of Applied Statistics*, 24(2):169–180, 1997. URL <https://doi.org/10.1080/02664769723783>. [p317]
- Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38(5):2916–2957, 2010. URL <https://doi.org/10.1214/10-AOS799>. [p305]
- M. J. Brewer. A Bayesian model for local smoothing in kernel density estimation. *Statistics and Computing*, 10(4):299–309, 2000. URL <https://doi.org/10.1023/A:1008925425102>. [p317]
- S. Calonico, M. D. Cattaneo, and M. H. Farrell. nprobust: Nonparametric kernel-based estimation and robust bias-corrected inference. *Journal of Statistical Software*, 91(8):1–33, 2019. URL <https://doi.org/10.18637/jss.v091.i08>. [p305]
- S. Cavuoti, V. Amaro, M. Brescia, C. Vellucci, C. Tortora, and G. Longo. Metaphor: a machine-learning-based method for the probability density estimation of photometric redshifts. *Monthly Notices of the Royal Astronomical Society*, 465(2):1959–1973, 2017. URL <https://doi.org/10.1017/S1743921317002186>. [p305]
- Y.-C. Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics and Epidemiology (Oxford, UK)*, 1(1):161–187, 2017. URL <https://doi.org/10.1080/24709360.2017.1396742>. [p305]
- H.-G. Drost. Philentropy: Information theory and distance quantification with R. *The Journal of Open Source Software*, 3:765, 2018. URL <https://doi.org/10.21105/joss.00765>. [p314]
- J. Farmer and D. Jacobs. High throughput nonparametric probability density estimation. *PLoS One*, 13(5):e0196937, 2018. URL <https://doi.org/10.1371/journal.pone.0196937>. [p305, 307]
- J. Farmer and D. J. Jacobs. *MATLAB tool for probability density assessment and nonparametric estimation*, 2022. URL <https://doi.org/10.1016/j.softx.2022.101017>. [p307]
- J. Farmer, Z. Merino, A. Gray, and D. Jacobs. Universal sample size invariant measures for uncertainty quantification in density estimation. *Entropy*, 21(11):1120, 2019. URL <https://doi.org/10.3390/e21111120>. [p306, 307]
- A. G. Gray and A. W. Moore. Nonparametric density estimation: Toward computational tractability. *Society for Industrial and Applied Mathematics. Proceedings of the SIAM International Conference on Data Mining*, pages 203–, 2003. URL <https://doi.org/10.1137/1.9781611972733.19>. [p305]
- T. Hayfield and J. S. Racine. Nonparametric econometrics: The np package. *Journal of Statistical Software*, 27(5), 2008. URL <https://doi.org/10.18637/jss.v027.i05>. [p305, 313, 314]
- A. J. Izenman and C. J. Sommer. Philatelic mixtures and multimodal densities. *Journal of the American Statistical Association*, 83(404):941–953, 1988. URL <https://doi.org/10.2307/2290118>. [p316, 317]
- D. J. Jacobs. Best probability density function from limited sampling. *Entropy*, 11:1001–1024, 2008. URL <https://doi.org/10.3390/e11041001>. [p308, 309]
- F. Konietzschke, M. Placzek, F. Schaarschmidt, and L. A. Hothorn. nparcomp: An R software package for nonparametric multiple comparisons and simultaneous confidence intervals. *Journal of Statistical Software, Articles*, 64(9):1–17, 2015. URL <https://doi.org/10.18637/jss.v064.i09>. [p305]
- T. Mildenerger and H. Weinert. The benchden package: Benchmark densities for nonparametric density estimation. *Journal of Statistical Software, Articles*, 46(14):1–14, 2012. URL <https://doi.org/10.18637/jss.v046.i14>. [p313, 316]
- J. Moss and M. Tveten. kdensity: An R package for kernel density estimation with parametric starts and asymmetric kernels. *Journal of Open Source Software*, page 1566, 2019. URL <https://doi.org/10.21105/joss.01566>. [p305]

- J. Munkhammar, L. Mattsson, and J. Ryden. Polynomial probability distribution estimation using the method of moments. *PLoS One*, 12(4):e0174573, 2017. URL <https://doi.org/10.1371/journal.pone.0174573>. [p305]
- T. Nagler and T. Vatter. *kde1d: Univariate Kernel Density Estimation*, 2020. URL <https://CRAN.R-project.org/package=kde1d>. R package version 1.0.3. [p305]
- P. Puchert, P. Hermosilla, T. Ritschel, and T. Ropinski. Data-driven deep density estimation. *Neural Computing and Applications*, 33:1–35, 2021. URL <https://doi.org/10.1007/s00521-021-06281-3>. [p307, 314]
- C. Sebastian, D. C. Matias, and H. F. Max. nprobust: Nonparametric kernel-based estimation and robust bias-corrected inference. *Journal of Statistical Software*, 91(1):1–33, 2019. URL <https://doi.org/10.18637/jss.v091.i08>. [p305]
- S. J. Sheather. The performance of six popular bandwidth selection methods on some real data sets. *Computational Statistics*, 7:225–250, 1992. [p317]
- A. Sidibé, G. Shu, Y. Ma, and W. Wanqi. Big data framework for abnormal vessel trajectories detection using adaptive kernel density estimation, 2018. URL <https://doi.org/10.1145/3291801.3291816>. [p305]
- L. Tang, Z. Kan, X. Zhang, F. Sun, X. Yang, and Q. Li. A network kernel density estimation for linear features in space-time analysis of big trace data. *International Journal of Geographical Information Science: Human Dynamics in the Mobile and Big Data Era*, 30(9):1717–1737, 2016. URL <https://doi.org/10.1080/13658816.2015.1119279>. [p305]
- M. Wand and B. Ripley. *KernSmooth: Functions for Kernel Smoothing Supporting Wand & Jones (1995)*, 2021. URL <https://CRAN.R-project.org/package=KernSmooth>. R package version 2.23-16. [p305, 313]
- M. P. Wand. *densEstBayes: Density Estimation via Bayesian Inference Engines*, 2021. URL <https://CRAN.R-project.org/package=densEstBayes>. R package version 1.0-2. [p305]
- S. S. Wilks. Order statistics. *Bull. Amer. Math. Soc.*, 54(1):6–50, 1948. [p306, 307]
- N. Wu. *The maximum entropy method*. Springer, NY, USA, 1997. [p307]

Jenny Farmer
Department of Bioinformatics
University of North Carolina at Charlotte
Charlotte, NC
United States of America
ORCID 0000-0002-7953-1044
jfarmer@carolina.rr.com

Donald Jacobs
Department of Physics and Optical Science
University of North Carolina at Charlotte
Charlotte, NC
United States of America
ORCID 0000-0001-7711-1639
djacobs1@uncc.edu

reclin2: a Toolkit for Record Linkage and Deduplication

by D. Jan van der Laan

Abstract The goal of record linkage and deduplication is to detect which records belong to the same object in data sets where the identifiers of the objects contain errors and missing values. The main design considerations of **reclin2** are: modularity/flexibility, speed and the ability to handle large data sets. The first points makes it easy for users to extend the package with custom process steps. This flexibility is obtained by using simple data structures and by following as close as possible common interfaces in R. For large problems it is possible to distribute the work over multiple worker nodes. A benchmark comparison to other record linkage packages for R, shows that for this specific benchmark, the **fastLink** package performs best. However, this package only performs one specific type of record linkage model. The performance of **reclin2** is not far behind the of **fastLink** while allowing for much greater flexibility.

1 Introduction

Combining different data sets is often an important step in many data analysis projects. Sometimes the data sets will contain high quality linkage keys, especially when the data sets are based on a common register. For example, the samples for (nearly) all social surveys performed at Statistics Netherlands are drawn from the population register and therefore can be linked to each other (Bakker et al., 2014). In these cases exact linkage can be used. In exact linkage, records are linked when they agree exactly on the linkage keys used. Exact linkage can be performed in R using base functions such as `merge`. However, it is not uncommon that data sets have to be linked on keys such as ‘first name’, ‘last name’ and ‘address’. Often these variables contain errors and/or missing values and, therefore, exact linkage is not possible. That is where probabilistic record linkage methods come into play (Herzog et al., 2007; Christen, 2012). These methods will calculate some sort of likelihood that two records belong to the same object (person, company, ...). This will be called a match. Only record pairs with a high enough likelihood are linked to each other. The goal is to minimise the number of false links (linking two records that do not belong to the same object) and the number of missed links (*not* linking two records that do belong to the same object).

The process of probabilistic record linkage generally consists of the following steps: (1) Generate pairs of records from each of the two data sets that are to be linked; (2) Compare the two records of the pair and generate a comparison vector (in the simplest case this a vector of ones and zeros coding agreement/disagreement on each of linkage keys); (3) Estimate a model that predicts based on the comparison vector a likelihood that the two records belong to the same object; (4) Select pairs with a high enough likelihood; (5) Using the selected pairs, generate the final linked dataset. **reclin2** offers different methods for most of these steps and by mixing the different methods a custom linkage process can be developed. This is discussed in more detail with examples in the section on the record linkage process.

A variant of record linkage is deduplication. Here there is only one data set and one wants to determine which records belong to the same object. For example, a customer database can contain the same customer multiple times with slightly different information (e.g. different email addresses). Deduplication is usually performed by linking a dataset to itself. Matches are then duplicate records. The principles are, therefore, the same as with regular record linkage and in the remainder of paper we will focus on regular record linkage of two data sets.

Record linkage can be computationally and memory intensive. In principle each record from a data set has to be compared to each record in the other data set. Therefore, when the two data sets are of size N_1 and N_2 respectively the computational complexity and memory requirements are of order $O(N_1N_2)$. For example, at Statistics Netherlands one common data set is the population register containing in the order of 10^7 records; other data sets can be in the order of 10^3 – 10^5 , resulting in 10^{10} – 10^{12} possible comparisons.

reclin2 is a package that provides a set of tools to perform probabilistic record linkage. It is the successor of the **reclin** package. The reason for the update was to be able to provide better support for the core design considerations of the **reclin/reclin2** package. Unfortunately this was not possible while keeping backward compatibility, therefore it was decided to continue with a new package. The core design considerations are:

1. Modularity/flexibility.

2. Speed.
3. Ability to handle large datasets.

The last two points are important because of the aforementioned issues with the size of the problem. The first point is important, as in practice no record linkage project is the same and, therefore, a common need is to vary on the default procedure. The next section will discuss how we tried to address the points above. Besides **reclin2** other packages exist for probabilistic record linkage. There is the **RecordLinkage** (Sariyar and Borg, 2022) package that implements various methods such as classic probabilistic record linkage based on the Fellegi and Sunter (1969) model and methods based on machine learning. Furthermore, there is the **fastLink** (Enamorado et al., 2020, 2019) package that focuses on a fast and flexible implementation of the Fellegi-Sunter model. The main difference of **reclin2** with these packages is the focus on the previous three points: **fastLink** scores well on points 2 and 3, but only supports one type of model while **RecordLinkage** scores well on point 3 and better than **fastLink** on point 1, but lacks some flexibility and speed. Points 2 and 3 are investigated in a later section using a benchmark.

2 Design considerations

One of the main considerations when designing the package was flexibility. Therefore, the package has been designed as a set of functions that operate on **data.table** objects (Dowle and Srinivasan, 2021). The main object of the package is the **pairs** object which is a subclass of **data.table**. The **pairs** object contains pairs of records from the two datasets that are to be linked (called *x* and *y*). The first two columns of the **pairs** object contain the indices of the corresponding records from the two data sets. Most functions of the package accept a **pairs** object and return a **pairs** object. The package has functions for different steps in the linkage process (as described in the next section). By combining the different available functions a custom data linkage process can be built. Furthermore, as the **pairs** object is a **data.table** it is also easy for the user to manipulate it. For example, new columns can be derived and **pairs** can be filtered. Functions that do not manipulate the **pairs** object are designed to follow as closely as possible the common interfaces of base R functions. For example, the function **problink_em** that can be used to estimate the parameters of the Fellegi-Sunter model accepts similar input as other modelling functions in R: e.g. a formula to specify the model and a **data** argument to pass in the data on which to estimate the model. The corresponding **predict** function can be used to calculate likelihoods for pairs being a match. It is therefore also easy to use models from other packages, such as machine learning methods, to estimate the likelihoods. Where a package such as **RecordLinkage** has functions for a number of machine learning methods, **reclin2** does not need these as the user is free to call these themselves as demonstrated in the section on the record linkage process below.

The other two design considerations, speed and being able to handle large datasets, are obtained in two ways. First, by using a **data.table** as the main object. Most methods have an **inplace** argument (default value is **FALSE**). When set to **TRUE** the **pairs** object is modified using the **[, :=]** operation of a **data.table**. This prevents unnecessary copies, decreasing memory consumption and increasing speed. Second, there is the option to create a cluster and distribute the computational load over multiple cores. Using functionality from the **parallel** or **snow** (Tierney et al., 2021) packages, multiple R processes are started and the data is distributed over these processes. Each process then generates a subset of the **pairs** which are kept within the process. Subsequent operations on the **pairs**, such as comparison, are also distributed over the processes where each process applies the operation to its subset of **pairs**. One of the more computationally intensive operations during linkage is comparing the records from the two data sets to each other. This problem scales well when parallelising. Therefore parallelization can lead to a significant speed up. Furthermore, when using a **snow** cluster the computation can also be distributed over multiple machines. This can not only lead to a speed up, but also means that the memory of multiple machines can be utilized allowing for larger problems than could be handled on a single machine. To work with a cluster, special functions with the **cluster_** prefix are offered. The cluster functions generating the **pairs** expect as one of their inputs a valid cluster created for example using **makeCluster** from the **parallel** package. When using the cluster variant, the object is no longer a **data.table** and it becomes more difficult to manually manipulate the object. The package has a few functions to help with this which will be discussed at the end of the next section.

3 The record linkage process

This section will give an overview of the linkage process and show how the functions in **reclin2** can be used for this. The discussion will be brief. An overview of the main steps of a record linkage

process has already been given in the introduction section of the paper. A more extensive description can be found in the package vignettes and documentation. Also, we will not go into detail into the methods used as these are well described in, for example, [Herzog et al. \(2007\)](#) and [Christen \(2012\)](#).

Generating pairs

The first step in the linkage process is to generate pairs of records from the two data sets x and y . There are a number of functions for this: the function `pair` generates all possible pairs. However, this can lead to impractically large numbers of pairs. Therefore, often methods are applied to reduce the total number of pairs. One commonly used method is blocking where only pairs are generated that agree on some key. This, of course, only works when a good enough quality key is available, otherwise true matches are lost. Another method in the package, `pairs_simsum`, is to generate pairs that agree on a given number of variables (e.g. they have to agree on either the postcode or the town name) ([Christen, 2012](#)). In the example below we use blocking on 'postcode', e.g. pairs are only generated when they agree exactly on 'postcode' ([...] in the examples indicate removed output).

```
> library(reclin2)
[...]
> data("linkexample1", "linkexample2")
> (pairs <- pair_blocking(linkexample1, linkexample2, "postcode"))
  First data set: 6 records
  Second data set: 5 records
  Total number of pairs: 17 pairs
  Blocking on: 'postcode'

  .x .y
1: 1 1
2: 1 2
3: 1 3
4: 2 1
5: 2 2
6: 2 3
7: 3 1
[...]
```

A `data.table` is returned with the added class `pairs`. The columns `.x` and `.y` contain the row indices into the two data sets. A copy (when the original data sets are not modified this is only a reference) of the two data sets is stored in the attributes `x` and `y`. This makes some of the next function calls easier.

There also exist cluster variants of these functions that return a `cluster_pairs` object:

```
> library(parallel)
> cl <- makeCluster(2)
> cpairs <- cluster_pair_blocking(cl, linkexample1, linkexample2, "postcode")
```

When calling the cluster variants of the pair generating algorithms, the records from x are randomly distributed over the nodes of the cluster and y is copied to each cluster node. On each node the corresponding pair function is called. The resulting pair object is stored on each node in an environment in the environment `reclin2::reclin_env` (the default name of this environment is "default"). The `cluster_pairs` object is a list with a copy of the cluster object and the name of the environment on the cluster nodes in which the pairs are stored.

Comparing pairs

The next step in the linkage process is to compare the pair of records on a set of common variables in both data sets. For this the package contains various comparison functions. The default function checks for exact agreement. However, for text fields such as names and addresses, it is often better to allow for spelling errors. For this some of the functions from the [stringdist \(van der Loo, 2014\)](#) package are imported. For classic record linkage using the Fellegi-Sunter model is necessary that these are translated into a similarity score between 0 and 1 where 1 is complete agreement which is what the functions included in `reclin2` do. In the example below, we provide a comparison function for 'firstname', 'lastname' and 'address':

```
> (compare_pairs(pairs, on = c("lastname", "firstname", "address", "sex"),
+   comparators = list(lastname = jaro_winkler(0.9), firstname = jaro_winkler(0.9),
```



```
+ address = jaro_winkler(0.9) ), inplace = TRUE))
[...]
```

	.x	.y	lastname	firstname	address	sex
1:	1	1	1.000000	0.472222	0.9230769	NA
2:	1	2	0.000000	0.583333	0.8641026	TRUE
3:	1	3	0.447619	0.4642857	0.933333	TRUE

```
[...]
```

The Jaro-Winkler string similarity score is used: a value of one indicates complete agreement, a value of zero indicated complete disagreement (no overlap in letters) and values in between indicate partial agreement. The 0.9 in the function call is a threshold used, among others, by the EM-algorithm discussed below as this method only handles complete agreement or disagreement: values above 0.9 are considered to agree completely. We see that the first record from x agrees exactly only on 'lastname' with the first record of y, while 'sex' cannot be compared as it is missing in at least one of the data sets.

The `compare_pairs` method is also implemented for the `cluster_pairs` object. For more flexibility there is also the `compare_vars` method. This function only compares one variable at the time, but it allows for different names of the variables in the two data sets, generating multiple output columns out of one comparison and for more complex comparisons where multiple variables are taken into account. As an example of the latter, the code below compares records on first name and last name allowing for the two parts of a name to be swapped:

```
> comp_name <- function(x, y) {
+   equal <- identical()
+   regular <- equal(x[[1]], y[[1]]) & equal(x[[2]], y[[2]])
+   swapped <- equal(x[[1]], y[[2]]) & equal(x[[2]], y[[1]])
+   regular | swapped
+ }
> compare_vars(pairs, "name_swap", on_x = c("firstname", "lastname"),
+   comparator = comp_name)
[...]
```

	.x	.y	lastname	firstname	address	sex	name_swap
1:	1	1	1.000000	0.472222	0.9230769	NA	FALSE
2:	1	2	0.000000	0.583333	0.8641026	TRUE	FALSE
3:	1	3	0.447619	0.4642857	0.933333	TRUE	FALSE

```
[...]
```

When records are compared on multiple columns, the comparison function receives two data.table objects as its inputs.

Scoring pairs

The goal of probabilistic record linkage is to generate a likelihood for each pair that the two records in the pair belong to the same record. This likelihood is based on the comparison vector. The traditional method is the model by [Fellegi and Sunter \(1969\)](#). The parameters of this model are usually estimated using a EM-algorithm ([Winkler, 2000](#)). However, `reclin2` considers this just a model as any other model and uses the same interface as any other model that can be estimated in R:

```
> m <- problink_em(~ lastname + firstname + address + sex, data = pairs)
> (pairs <- predict(m, pairs = pairs, add = TRUE))
[...]
```

	.x	.y	lastname	firstname	address	sex	weights
1:	1	1	1.000000	0.472222	0.9230769	NA	7.7103862
2:	1	2	0.000000	0.583333	0.8641026	TRUE	-5.9463949
3:	1	3	0.447619	0.4642857	0.933333	TRUE	0.8042090

```
[...]
```

The range of the weights depends on the number of variables and the estimated parameters in the model. They are log-likelihood ratios ([Fellegi and Sunter, 1969](#)). The values of the weights themselves are not directly of use, except that a higher weight indicates that a pairs if more likely a match. In principle, a weight above zero indicates that the pair is more likely a match than not. However, in practice, a threshold higher than zero is often used in order to reduce the likelihood of false links. The `predict` function of the EM-model also has the option to estimate posterior probabilities. Thresholds for the weights (or probabilities) are often determined by manually inspecting pairs around potential threshold values ([Herzog et al., 2007](#)). These methods can also be used for `cluster_pairs` objects.

As the `pairs` object is a regular data table object, it is also relatively easy to estimate other models on the data set. In principle this is a classification problem: the pairs need to be divided into two categories: matches and non-matches. For example, when for a part of the pairs the true match status is known, a supervised learning method can be used. In the example below the 'id' field is used to derive the true match status for the dataset (in practice this would probably only be available for a subset) and predict a linkage probability using logistic regression:

```
> compare_vars(pairs, "true", on_x = "id", on_y = "id", inplace = TRUE)
> mglm <- glm(true ~ lastname + firstname, data = pairs,
+ family = binomial())
> pairs[, pglm := predict(mglm, type = "response")]
```

In the first line of this example, a column named 'true' is added to the dataset (in place). This column is a comparison of the 'id' from the first data set (`on_x = "id"`) to the 'id' column of the second data set (`on_y = "id"`). This column 'true' contains the true match status. In the second line, a logistic regression model is estimated that predicts the match status using the two columns 'firstname' and 'lastname'. Using this model the probability of a true match is estimated in the final line of the example and added to the data set.

Creating the linked data set

In order to link the pairs a suitable threshold needs to be determined for the weights. Records with a weight above this threshold are classified as a match. Also, we generally know that each person only has one record in each data set. So, generally we will want to enforce one-to-one linkage. This will also generally improve the quality of the linked data set. `reclin2` has two methods for enforcing one-to-one linkage. The method `select_n_to_m` tries to select the pairs in such a way that the total weight of the selected pairs is maximised while linking each record from each data set to at most one record from the other data set (using its arguments it is also possible to enforce n-to-one or one-to-n linkage). A faster method that can lead to less links is `select_greedy` that will try to select the pair with the highest weight for each record. Below the first method is applied; records with a weight below 0 are not considered (`threshold = 0`):

```
> (pairs <- select_n_to_m(pairs, "weights", variable = "select", threshold = 0))
[...]
```

	.x	.y	lastname	firstname	address	sex	weights	select
1:	1	1	1.000000	0.4722222	0.9230769	NA	7.7103862	FALSE
2:	1	2	0.000000	0.5833333	0.8641026	TRUE	-5.9463949	FALSE
3:	1	3	0.447619	0.4642857	0.9333333	TRUE	0.8042090	FALSE
4:	2	1	1.000000	0.8888889	0.9230769	NA	8.6064218	TRUE

```
[...]
```

The method creates a logical column (name given by the 'variable' argument) in the `pairs` object with the selected pairs. The first pair has a high enough weight to be selected, but there is another candidate for the first record of `y` that is more likely, namely record 2 from `x` (see fourth row of the output above).

Up until now we are still working with a set of pairs. The goal is to get an actually linked dataset containing matched records from both data sets. This can be done using the `link` method. This function takes the `pairs` and the name of a logical column indicating which pairs are selected and it will generate the final linked data set. The output is similar to that of `merge`. The method also has arguments `all_x` and `all_y` that function the same as the corresponding `all.x` and `all.y` arguments of `merge`.

```
> (linked_data_set <- link(pairs, selection = "select"))
Total number of pairs: 4 pairs
```

	.y	.x	id.x	lastname.x	firstname.x	address.x	sex.x	postcode.x	id.y
1:	1	2	2	Smith	George	12 Mainstr	M	1234 AB	2
2:	2	3	3	Johnson	Anna	61 Mainstr	F	1234 AB	3
3:	3	4	4	Johnson	Charles	61 Mainstr	M	1234 AB	4
4:	4	6	6	Schwartz	Ben	1 Eaststr	M	6789 XY	6

	lastname.y	firstname.y	address.y	sex.y	postcode.y
1:	Smith	George	12 Mainstreet	<NA>	1234 AB
2:	Jonson	A.	61 Mainstreet	F	1234 AB
3:	Johnson	Charles	61 Mainstr	F	1234 AB
4:	Schwartz	Ben	1 Main	M	6789 XY

For the `cluster_pairs` the steps above need to change a little bit as `select_n_to_m` needs to consider all pairs and, therefore, does not work with objects of type `cluster_pair` where the pairs are distributed over the cluster nodes. Therefore, we first need to copy the relevant pairs to the main R process. We can use a selection variable for this only returning the pairs with a weight above zero:

```
> cpairs <- predict(m, pairs = cpairs, add = TRUE)
> select_threshold(cpairs, "weights", variable = "initial", threshold = 0)
> local_cpairs <- cluster_collect(cpairs, "initial")
> local_cpairs <- select_n_to_m(local_cpairs, "weights", variable = "select")
```

The first line calculates weights for the `cpairs` object. In the second line a logical column 'initial' is created which is TRUE for records with a weight higher than 0. Using `cluster_collect`, we collect the pairs from the worker processes into the main R process. Using the second argument we only collect pairs for which the column 'initial' is TRUE. The `local_cpairs` object is a regular pairs object (and, therefore, also a `data.table`) on which we can use the regular `select_n_to_m` method.

Helper functions for `cluster_pair` objects

It is easy to do manual manipulations on the regular pairs object. For the `cluster_pairs` the pairs are distributed over the worker nodes. There are a couple of functions to help with this. The already mentioned `cluster_collect` function copies the pairs locally. The `cluster_call` function accepts the `cluster_pairs` and a function. It will call the function on each node and pass it the `pairs`, `x` and `y`. The results of the functions are copied back locally. For example to get the number of pairs on each node:

```
> unlist(cluster_call(cpairs, \(p, ...) nrow(p)))
[1] 9 8
```

The `cluster_modify_pairs` function can be used to modify the pairs. The arguments are the `cluster_pairs` and a function with the same arguments as for `cluster_call`. The result of that function overwrites the `pairs` object on the worker node (except when NULL). In the example below, this is used to remove pairs with a weight of zero or lower.

```
> (cluster_modify_pairs(cpairs, \(p, ...) p[weights > 0, ]))
Cluster 'default' with size: 2
First data set: 6 records
Second data set: 5 records
Total number of pairs: 15 pairs
Blocking on: 'postcode'
```

Showing a random selection of pairs:

```
.x .y lastname firstname address sex weights initial
1: 3 1 0.447619 0.472222 0.8641026 NA 0.6017106 TRUE
2: 4 3 1.000000 1.000000 1.000000 FALSE 15.4915816 TRUE
[...]
```

Note, that the original `cpairs` object has been modified. Using the `new_name` argument it is also possible to generate a new set of pairs.

4 Benchmark

In this section the performance of the packages for data linkage will be investigated using example data from the Eurostat financed ESSnet (European Statistical System Centres and Networks of Excellence) project on Data Integration (Eurostat, 2011). The two data sets 'PDR' and 'CIS' were linked to each other. The datasets have 24,750 and 24,613 records respectively resulting in 612,562,500 possible pairs when linking the complete data sets. To study the effect of the size of the problem on the performance, samples were drawn from the two data sets based on the postcode where care was taken to sample the same postcodes in the two datasets. The sample fraction was varied from 0.1 to 1.0 in steps of 0.1.

The methods are compared on total computation time and memory use. These were measured using the 'time' program on a Linux server. The virtual server has 16 3.2GHz Intel Xeon Gold 6146 cores and 512GB of memory and runs on Ubuntu 20.04. For the time the reported 'elapsed (wall clock) time' is used and for the memory usage the 'maximum resident set size'. For `reclin2` also the effect of different numbers of worker nodes was investigated using the cluster functions. It was attempted to keep the methods used as close as possible to each other. No blocking was applied. The EM-algorithm

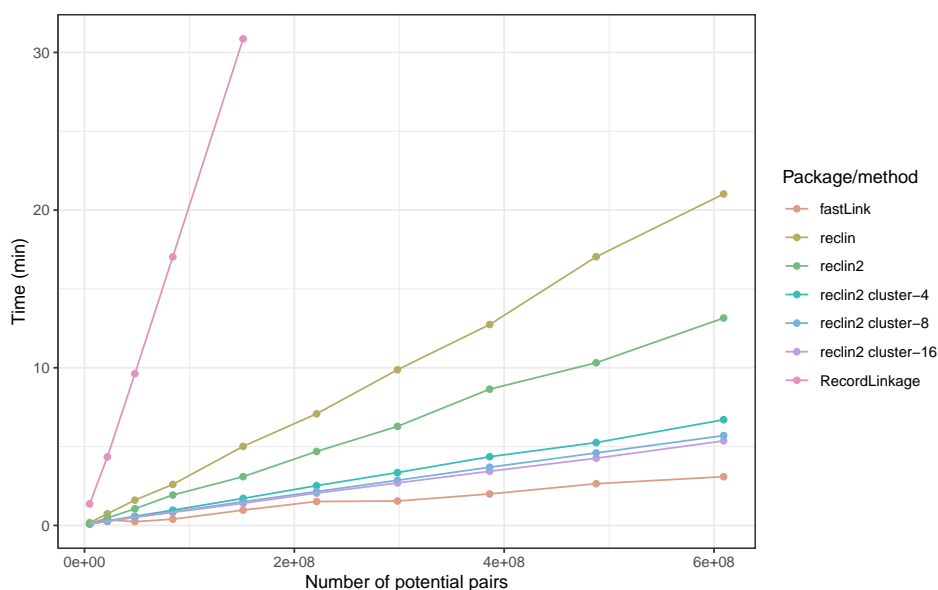


Figure 1: Comparison of computation times (in minutes) for the different packages (lines) as function of the number of potential pairs (the product of the sizes of the two data sets). For **reclin2** also different numbers of worker nodes were investigated; these are denoted by ‘cluster’ and the number of worker nodes.

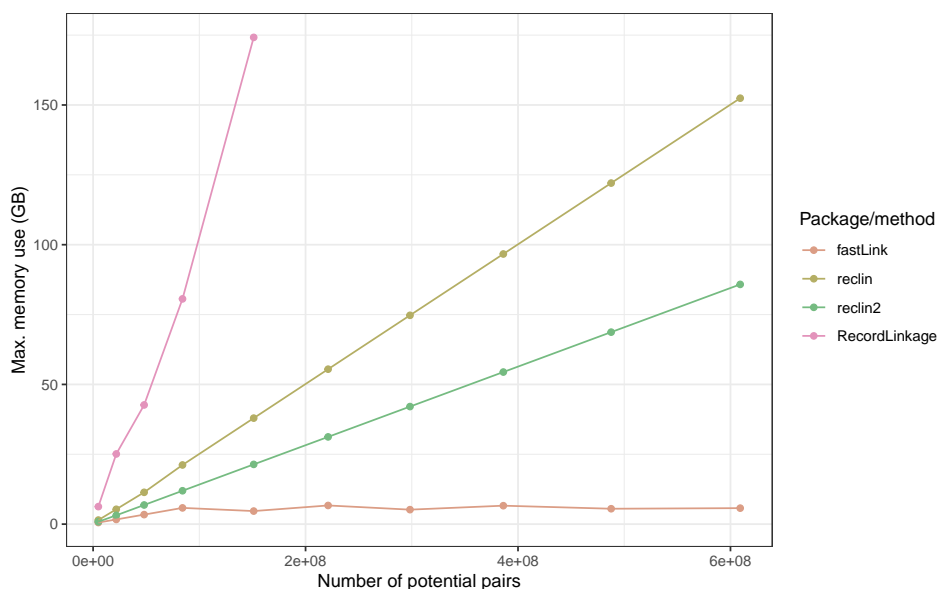


Figure 2: Comparison of memory usage (in gigabytes) for the different packages (lines) as function of the number of potential pairs (the product of the sizes of the two data sets). For **reclin2** no reliable estimates could be obtained for the runs with multiple workers. Therefore results are only presented for the single threaded version of **reclin2**. In principle the memory usage should not depend on the number of workers.

was used. For comparing the names the Jaro-Winkler similarity score was used with a threshold of 0.85. The quality of the resulting record linkage was recorded, but as the methods were not optimally tuned, it is difficult to compare these results and these results are, therefore, not reported. The complete code for the benchmark and all results can be found on Github (van der Laan, 2022).

Figures 1 and 2 show the computation times and the memory usage respectively as a function of the size of the problem. For the **RecordLinkage** package larger problem sizes than reported here were not investigated as the system started running out of memory. **RecordLinkage** has the option to also work from disk for large problems. The performance of this was not investigated as this would only lead to longer computation times which were already longer than those of the other packages. For this specific problem the **fastLink** package performs better than the other packages. Especially the memory usage is substantially lower because of their use of specialised data structures (Enamorado et al., 2019). The difference in computation time between **fastLink** and **reclin2** using multiple cores is limited (approx. factor 1.7 for the largest problem). However, **fastLink** is tailored for use with the EM-algorithm while the other packages are more general.

The speed-up of the **reclin2** benchmark is not proportional to the number of cores used. This is caused mainly by the fact that some of the steps take place in the main process: reading and sampling the data, starting the worker nodes, and importantly, finalizing the record linkage using one-to-one matching. We are unfortunately limited by Amdahl's law (Amdahl, 1967) although the generation, comparison and (in case of the EM-algorithm) tabulation of pair and calculating the predictions will take an increasingly larger part of the running time as the size of problem increases.

5 Conclusion

By using simple data structures, namely `data.table` objects, and providing a set of functions that operate on these structures, we have built a flexible and well performing toolkit for record linkage. For users, it is easy to extend on the methods present in the package. Either by manipulating the data structures directly, by writing custom functions or by using existing functions. An example of the latter, is the relative ease with which existing machine learning methods can be used in the record linkage process.

The package manages to keep memory use limited and on machines with 256GB of memory it should handle problems up to approximately 10^9 pairs. By combining the memory of multiple machines this can of course be extended. When one is only interested in using the Fellegi-Sunter model of record linkage with an EM-algorithm to estimate the parameters of that model, the **fastLink** package is probably the best choice. It performs better and the model and EM-algorithm used is more flexible than that currently present in **reclin2**. The main advantage of **reclin2** over **fastLink** is the flexibility **reclin2** provides. Especially for non standard problems this is important.

The package is still being developed. One of the things that is being worked on is the option to take into account the uniqueness of a certain attribute. For example, agreement on a rare family name is a stronger indication of a match than agreement on a common family name. However, we hope that making it easy for users to extend and modify the record linkage processes also lowers the threshold for contributing to the package.

Bibliography

- G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, AFIPS '67* (Spring), pages 483–485, New York, NY, USA, 1967. Association for Computing Machinery. URL <https://doi.org/10.1145/1465482.1465560>. [p327]
- B. F. M. Bakker, J. van Rooijen, and L. van Toor. The system of social statistical datasets of statistics netherlands: An integral approach to the production of register-based social statistics. *Journal of the International Association for Official Statistics*, 30:1–14, 2014. URL <https://doi.org/10.3233/SJI-140803>. [p320]
- P. Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution and Duplicate Detection*. Springer-Verlag, Berlin, 2012. [p320, 322]
- M. Dowle and A. Srinivasan. *data.table: Extension of 'data.frame'*, 2021. URL <https://CRAN.R-project.org/package=data.table>. R package version 1.14.0. [p321]

- T. Enamorado, B. Fifield, and K. Imai. Using a probabilistic model to assist merging of large-scale administrative records. *American Political Science Review*, 113(2):353–371, 2019. URL <https://doi.org/10.1017/S0003055418000783>. [p321, 327]
- T. Enamorado, B. Fifield, and K. Imai. *fastLink: Fast Probabilistic Record Linkage with Missing Data*, 2020. URL <https://CRAN.R-project.org/package=fastLink>. R package version 0.6.0. [p321]
- Eurostat. ESSnet DI: fictitious data from ons for on-the-job training on record linkage, 2011. URL https://ec.europa.eu/eurostat/cros/content/essnet-di-fictitious-data-ons-job-training-record-linkage_en. Accessed: 2022-01-13. [p325]
- I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969. URL <https://doi.org/10.1080/01621459.1969.10501049>. [p321, 323]
- T. N. Herzog, F. J. Scheuren, and W. E. Winkler. *Data Quality and Record Linkage Techniques*. Springer, New York, 2007. [p320, 322, 323]
- M. Sariyar and A. Borg. *RecordLinkage: Record Linkage Functions for Linking and Deduplicating Data Sets*, 2022. URL <https://CRAN.R-project.org/package=RecordLinkage>. R package version 0.4-12.3. [p321]
- L. Tierney, A. J. Rossini, N. Li, and H. Sevcikova. *snow: Simple Network of Workstations*, 2021. URL <https://CRAN.R-project.org/package=snow>. R package version 0.4-4. [p321]
- D. J. van der Laan. *reclin2-benchmark*, 2022. URL <https://github.com/djvanderlaan/reclin2-benchmark>. Git repository with code used for the benchmark. Tag: v1.0. [p327]
- M. van der Loo. The stringdist package for approximate string matching. *The R Journal*, 6:111–122, 2014. URL <https://CRAN.R-project.org/package=stringdist>. [p322]
- W. E. Winkler. *Using the EM algorithm for weight computation in the Fellegi-Sunter model of record linkage*. US Bureau of the Census Washington, DC, 2000. [p323]

D. Jan van der Laan
Statistics Netherlands (CBS)
Henri Faasdreef 313, The Hague
The Netherlands
ORCID: 0000-0002-0693-1514
dj.vanderlaan@cbs.nl

News from the Bioconductor Project

by *Bioconductor Core Team*

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. The project has entered its twentieth year, with funding for core development and infrastructure maintenance secured through 2025 (NIH NHGRI 2U24HG004059). Additional support is provided by NIH NCI, Chan-Zuckerberg Initiative, National Science Foundation, Microsoft, and Amazon. In this news report, we give some details about the software and data resource collection, infrastructure for building, checking, and distributing resources, core team activities, and some new initiatives.

Software ecosystem

Bioconductor 3.15 was released on 27 April, 2022. It is compatible with R 4.2.0 and consists of 2140 software packages, 410 experiment data packages, 990 up-to-date annotation packages, 29 workflows, and 3 books. **Books** are built regularly from source and therefore fully reproducible; an example is the community-developed [Orchestrating Single-Cell Analysis with Bioconductor](#). The [Bioconductor 3.15 release announcement](#) includes descriptions of 78 new software packages, and updates to NEWS files for many additional packages.

Infrastructure updates

- Thanks to a generous allocation (BIR190004, "Engineering and disseminating a software and analysis ecosystem for genomic data science") provided through the National Science Foundation ACCESS (formerly XSEDE) program, academic cloud resources including GPUs and highly accessible object storage systems are being integrated into project operations.
- Transition of primary funding administration from Roswell Park Comprehensive Cancer Center to Dana-Farber Cancer Institute has led to a number of changes to platforms in use for the checking and production of binary package images.
 - Linux builds occur at Dana-Farber Cancer Institute.
 - Windows builds occur in machinery provided by Microsoft Genomics in the Azure cloud environment.
 - MacOS builds occur at Dana-Farber Cancer Institute. Work on the support of ARM Mac systems occurs at MacStadium.
 - Details on the configurations of builders (e.g., [the Linux builder](#) for the devel branch) are available at the [Build reports](#) link at [bioconductor.org](#).
- An interactive app for surveying adverse conditions arising for package install, build, and check processes has been introduced for [release](#) and [devel](#) branches.
- Cloud-based workshop delivery systems have been an integral part of Bioconductor conferences and teaching activities.
 - Workshops from Bioconductor 2022 are continuously available for inspection and hands-on exercises at <http://app.orchestra.cancerdatasci.org>, thanks to cloud computing support provided by Dr. Sean Davis of University of Colorado.
 - <http://workshop.bioconductor.org> is a Galaxy-based workshop collection deployed on Jetstream2 in NSF ACCESS.

Core team updates

- After six years of highly effective work in the core, Nitesh Turaga has left for a position in industry. We will miss him!
- New core developers Jen Wokaty and Alexandru Mahmoud have joined. Jen is a member of the Waldron Lab at CUNY. Alex works at Channing Division of Network Medicine.

- Jen and Alex are joined by long-term core members Lori Kern of Roswell Park Comprehensive Cancer Center, Marcel Ramos of CUNY and Roswell, and Hervé Pages of Fred Hutchinson Cancer Research Center.

New initiatives

- Thanks to efforts of members of the Technical and Community Advisory Boards and community members, a collection of working groups has been defined to achieve new project aims. An [overview](#) of currently active working groups is available, along with [guidelines for proposing new working groups](#).
- The objectives of the bioconductor-teaching working group are stated at the associated [repository](#):

The Bioconductor teaching committee is a collaborative effort to consolidate Bioconductor-focused training material and establish a community of Bioconductor trainers. We define a curriculum and implement online lessons for beginner and more advanced R users who want to learn to analyse their data with Bioconductor packages.

- A [mentoring program](#) for new developers has taken flight.
- Thanks to an Essential Open Source Software grant from the Chan-Zuckerberg Initiative, we have partnered with the Dana-Farber Cancer Institute [YES for CURE](#) (Young Empowered Scientists for Continued Research Engagement) program to offer instruction in cancer data science to interested undergraduates. A [pkgdown site](#) includes current curricular materials.
- With the NSF-based academic cloud resources previously mentioned, we have begun gestation of G-DADS, a program for Genomic Data and Analysis Development Services, with the objectives of providing publicly accessible storage and compute on exemplars of the latest high-volume experimental modalities, and of promoting GPUs to first-class citizenship in our build and check systems.

Using Bioconductor

Start using Bioconductor by installing the most recent version of R and evaluating the commands

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install()
```

Install additional packages and dependencies, e.g., [SingleCellExperiment](#), with

```
BiocManager::install("SingleCellExperiment")
```

[Docker](#) images provides a very effective on-ramp for power users to rapidly obtain access to standardized and scalable computing environments. Key resources include:

- [bioconductor.org](#) to install, learn, use, and develop Bioconductor packages.
- A list of [available software](#), linking to pages describing each package.
- A question-and-answer style [user support site](#) and developer-oriented [mailing list](#).
- A community slack ([sign up](#)) for extended technical discussion.
- The [F1000Research Bioconductor channel](#) for peer-reviewed Bioconductor work flows.
- The [Bioconductor YouTube](#) channel includes recordings of keynote and talks from recent conferences including Bioc2022, EuroBioC2022, and BiocAsia2021, in addition to video recordings of training courses.

- Our [package submission](#) repository for open technical review of new packages.

Recent Bioconductor conferences include [BioC 2022](#) (July 27-29), and [European Bioconductor Meeting](#) (September 14-16). Each had invited and contributed talks, as well as workshops and other sessions to enable community participation. Slides, videos, and workshop material for each conference are, or will soon be, available on each conference web site as well as from the [Courses and Conferences](#) section of the Bioconductor web site.

The Bioconductor project continues to mature as a community. The [Technical](#) and [Community](#) Advisory Boards provide guidance to ensure that the project addresses leading-edge biological problems with advanced technical approaches, and adopts practices (such as a project-wide [Code of Conduct](#)) that encourages all to participate. We look forward to welcoming you!

Bioconductor Core Team
Channing Division of Network Medicine
Mass General Brigham
Harvard Medical School, Boston, MA

Department of Data Science
Dana-Farber Cancer Institute
Harvard Medical School, Boston, MA

Biostatistics and Bioinformatics
Roswell Park Comprehensive Cancer Center, Buffalo, NY

Fred Hutchinson Cancer Research Center, Seattle, WA

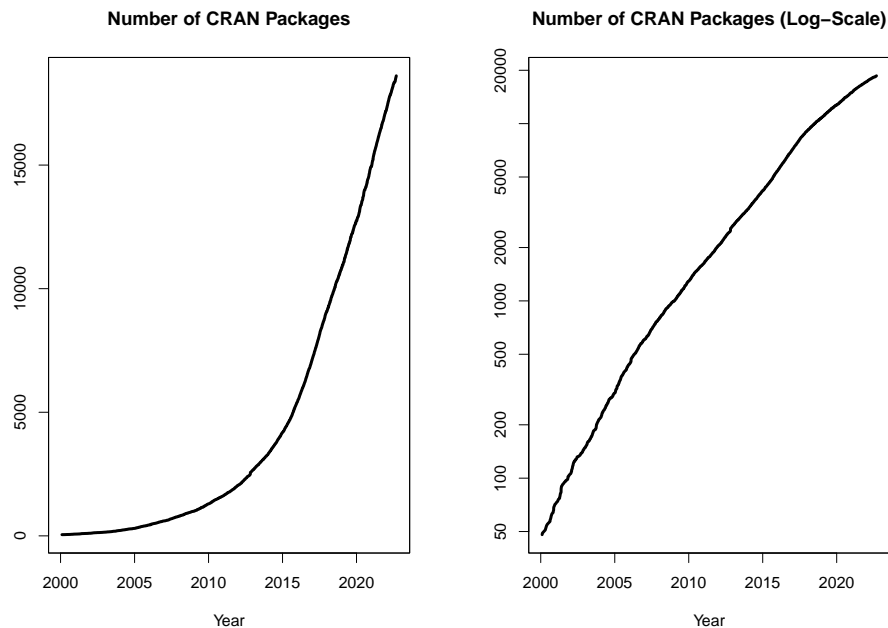
CUNY Graduate School of Public Health, New York, NY
maintainer@bioconductor.org

Changes on CRAN

2022-04-01 to 2022-06-30

by Kurt Hornik, Uwe Ligges and Achim Zeileis

In the past 3 months, 485 new packages were added to the CRAN package repository. 76 packages were unarchived, 1179 were archived and 3 had to be removed. 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 18260.

Changes in the CRAN Repository Policy

The [Policy](#) now says the following:

- The ownership of copyright and intellectual property rights of all components of the package must be clear and unambiguous [...] ('All components' includes any downloaded at installation or during use.)
- The package's 'DESCRIPTION' file must show both the name and email address of a single designated maintainer (a person, not a mailing list). That contact address must be kept up to date, and be usable for information mailed by the CRAN team without any form of filtering, confirmation . . . Forwarding mail from the maintainer address increasingly results in confusing non-delivery notifications to the original sender, so is best avoided.
- Security provisions must not be circumvented, for example by not verifying SSL/TLS certificates.

[External Libraries for CRAN packages](#) now says

- For macOS: JAGS may or may not be available. There is an 'official' release for both architectures at <https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/>.
- For Windows: The build system for Windows changed with 4.2.0 and only that is considered here (and only 64-bit Windows is now supported).

CRAN package submissions

In the second third of 2022 (May 2022 to August 2022), CRAN received 9860 package submissions. For these, 17476 actions took place of which 11354 (65%) were auto processed actions and 6122 (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	2559	2660	1635	0	0	2830	933	737
manual	2178	113	568	255	55	2226	595	132

These include the final decisions for the submissions which were

action	archive	publish
auto	2462 (25.5%)	2311 (23.9%)
manual	2156 (22.3%)	2735 (28.3%)

where we only count those as *auto* processed whose publication or rejection happened automatically in all steps.

A new team member, Benjamin Altmann, joined the CRAN submission team. Welcome, Beni. Unfortunately, Gregor Seyer left the CRAN submission team after processing 5482 incoming submissions. Thanks a lot!

CRAN mirror security

Currently, there are 102 official CRAN mirrors, 80 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

There are three new task views:

CausalInference Maintained by Imke Mayer, Pan Zhao, Noah Greifer, Nick Huntington-Klein, and Julie Josse.

Epidemiology Maintained by Thibaut Jombart, Matthieu Rolland, and Hugo Gruson.

SportsAnalytics Maintained by Benjamin S. Baumer, Quang Nguyen, and Gregory J. Matthews.

Currently there are 39 task views (see <https://cran.r-project.org/web/views/>), with median and mean numbers of CRAN packages covered 101 and 112, respectively. Overall, these task views cover 3668 CRAN packages, which is about 20% of all active CRAN packages.

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

News from the Forwards Taskforce

by Heather Turner

[Forwards](#) is an R Foundation taskforce working to widen the participation of under-represented groups in the R project and in related activities, such as the *useR!* conference. This report rounds up activities of the taskforce during the first half of 2022.

Accessibility

As another step towards improving the accessibility of the R Journal, Di Cook and Heather Turner are mentoring a Google Summer of Code student, Abhishek Ulayil, on the project [Converting past R Journal articles to HTML](#). This project has benefited from regular input from Mitchell O'Hara Wild and Christophe Dervieux, authors of the `rjtools` package that provides the new HTML template for R Journal articles (O'Hara-Wild et al. 2022).

s gwynn sturdevant and Jonathan Godfrey were part of an invited panel at *JSM 2022* on [Delivering Data Differently](#) that explored alternatives to data visualisation.

Community engagement

The community team have taken a number of actions to support the R community in Africa. A WhatsApp group has been set up for leaders of African R User Groups, to facilitate collaboration. Kevin O'Brien has been a Zoom host for several R User Groups, including the Botswana, Eswatini and Bulawayo groups. Zane Dax worked with the Accra R User Group on graphical design for advertising their meetups. Kevin O'Brien and Sam Toet helped to organize the first [Francophone satRday](#), which featured a line up of African speakers.

Another focus has been the relaunch of [RainbowR](#) led by Ella Kaye and Zane Dax. Following a well-attended online meetup, the website was rebuilt, the Slack group opened to new members with a new code of conduct, and the Twitter account has been in active use. The group plan to have regular online meetups and to raise awareness of issues affecting the LGBTQ+ community through sharing relevant data sets for exploration and teaching.

Beyond this, taskforce members continue to engage with a range of communities. Zane Dax contributed to an update of the [Minorities in R \(MiR\)](#) website. Yanina Bellini Saibene and Heather Turner joined the [Building inclusive communities](#) panel at the launch of the AsiaR community, to share their experience from working with different communities.

Conferences

Yanina Bellini Saibene assisted the *useR! 2022* team on behalf of the R Foundation, to share expertise in the organization of virtual conferences and help incorporate good practices that encourage diverse participation. (The conference was originally planned to be hybrid, but moved to be completely online.) Such practices included adding representatives from different regions to the organizing team, securing funding to caption talks, accepting elevator pitches and tutorials in non-English languages, and adjusting the registration fees to the income group of each participant's country of residence. Many of these practices were based on the work of the organizing team of *useR! 2021* - that included Yanina and other Forwards members - who summarized their recommendations in the recent publication: "Ten simple rules to host an inclusive conference" (Joo et al. 2022).

R Contribution

Saranjeet Kaur Bhogal and Heather Turner organized a series of [Collaboration Campfires](#) with a goal to demystify the R development process and highlight ways that R programmers can contribute. The first two sessions explored R's bug-tracking process and how R users can

contribute to reviewing bugs. The second two sessions explored R's process for localization and how to contribute to a translation team. The sessions attracted a diverse group of participants who engaged with the interactive activities, providing a foundation for further engagement.

The R Contribution working Group (RCWG) organized a [Bug BBQ](#) as a satellite to *useR!* 2022. Members of the RCWG prepared a number of open bugs in advance, for participants to look at in one or more of three organized online sessions. The event was supported by several R Core members and attended by both novice and experienced contributors. For experienced contributors the event provided a spur to work on open bugs, leading to progress on several issues. Meanwhile novice contributors contributed to the analysis of open bugs, under the guidance of experienced contributors. As the first event of its kind, the event showed promise as a way to engage the wider R community in contribution.

Saranjeet Kaur Bhogal has been working on a new chapter for the [R Development Guide](#) on contributing translations, as part of the R project's [Google Summer of Docs project](#), with substantial contribution from Michael Chirico. Along with Ben Ubah, Michael is co-mentoring a Google Summer of Code student, Meet Bhatnagar, to create a dashboard to [monitor the status of translations in R](#).

Changes in Membership

New members

We welcome the following member to the taskforce:

- Community team: Ella Kaye (co-leader)

Previous members

The following members have stepped down:

- Community team: Richard Ngamita (co-leader)
- Accessibility team: Becca Wilson
- Surveys team: Anna Vasylytsya (co-leader)

We thank them for their contribution to the taskforce.

References

- Joo, Rocío, Andrea Sánchez-Tapia, Sara Mortara, Yanina Bellini Saibene, Heather Turner, Dorothea Hug Peter, Natalia Soledad Morandeira, et al. 2022. "Ten Simple Rules to Host an Inclusive Conference." *PLOS Computational Biology* 18 (7): 1–13. <https://doi.org/10.1371/journal.pcbi.1010164>.
- O'Hara-Wild, Mitchell, Stephanie Kobakian, H. Sherry Zhang, Di Cook, and Christophe Dervieux. 2022. *rjtools: Tools for Preparing, Checking, and Submitting Articles to the R Journal*. <https://github.com/rjournal/rjtools>.

Heather Turner
University of Warwick
UK
Heather.Turner@R-Project.org

Changes in R

by Tomas Kalibera, Sebastian Meyer, Kurt Hornik

Abstract We give a selection of the most important changes in R 4.2.0 and of subsequent bug fixes for the Windows port of R. We also provide statistics on source code commits.

1 R 4.2.0 selected changes

R 4.2.0 (codename “Vigorous Calisthenics”) was released on 2022-04-22. The December 2021 (13/2) issue of the R Journal provided a selection of the most important changes in that release that were already settled at that time, including:

- R on Windows uses UTF-8 as the native encoding, uses the new Universal C Runtime (UCRT) and a new toolchain (Rtools42).
- R on Windows changed the default library location and the default installation location for user-only installation to match current Windows conventions.
- Support for isolated groups, compositing operators, affine transformations, and stroking and filling paths has been added to the R graphics engine.
- R now provides an R-level interface for hash tables.

A selection of the remaining important R 4.2.0 changes is provided here.

- The HTML help system has several new features: \LaTeX -like math can be typeset using either [KaTeX](#) or [MathJax](#), usage and example code is highlighted using [Prism](#), and for dynamic help the output of examples and demos can be shown within the browser if the [knitr](#) package is installed. (These features can be disabled by setting the environment variable `_R_HELP_ENABLE_ENHANCED_HTML_` to a false value.)

The HTML help system now uses HTML5, the current HTML standard, which in particular helps to facilitate some of the enhancements described above. Considerable effort was put into ensuring *valid* HTML5 output. The old validation toolchain could not handle HTML5, so a new one was created based on [HTML Tidy](#) and integrated into the `tools` package. R CMD check can now optionally (but included in ‘--as-cran’) validate the package HTML help files.

See <https://blog.r-project.org/2022/04/08/enhancements-to-html-documentation/> for more information.

- Calling `if()` or `while()` with a condition of length greater than one now gives an error rather than a warning. Consequently, environment variable `_R_CHECK_LENGTH_1_CONDITION_` no longer has any effect. Similarly, calling `&&` or `||` with either argument of length greater than one now gives a warning. In R 4.3.0, it will give an error, and environment variable `_R_CHECK_LENGTH_1_LOGIC2_` will no longer have any effect.
- The `grid` package now allows the user to specify a *vector* of pattern fills. The `fill` argument to `gpar()` accepts a list of gradients and/or patterns and the functions `linearGradient()`, `radialGradient()`, and `pattern()` have a new `group` argument. Finally, points grobs (data symbols) can now also have a pattern fill.
See <https://blog.r-project.org/2022/06/09/vectorised-patterns-in-r-graphics/> for more information.
- In `lhs |> rhs` expressions using the native pipe operator it is now possible to use a named argument with the placeholder `_` in the `rhs` call to specify where the `lhs` is to be inserted. The placeholder can only appear once on the `rhs`.
- On Windows, `download.file(method = "auto")` and `url(method = "default")` now follow Unix in using “libcurl” for all except ‘file://’ URIs. This impacts, for example, updating of R packages (HTTPS downloads). Most users should not notice, but an additional proxy setup may be required (see `help("download.file")`). Also, “libcurl” is by the decision of the library authors stricter in checking certificate revocation than the previous “wininet” method. This brings more security, but also may cause trouble with some corporate HTTPS MITM proxies, which filter HTTPS traffic. R-patched (to become R 4.2.2) has a work-around via environment variable `R_LIBCURL_SSL_REVOKE_BEST_EFFORT`.

2 R 4.2.1 and R-patched changes on Windows

R 4.2.0 on Windows switched to UTF-8 as the native encoding and to UCRT as the Windows runtime. R is an early adopter of UCRT in the open-source community of projects compiled using free and open-source compilers, and particularly an early adopter of UTF-8 as the native encoding on Windows, hence this came with considerable effort and risk of bugs. Hence, testing using CRAN package checks has been in place for over a year before the release (and 9 months of that time in parallel to the usual CRAN checks when R-devel still used MSVCRT as the C runtime).

Still, some issues not covered by such tests, mostly in interactive use and Rgui, have been reported by users after the 4.2 release and have been fixed in R-patched. R users on Windows should update to the latest available patch version.

Selected fixes in R 4.2.1:

- Accent keys now work in GraphApp Unicode windows, which are used by Rgui whenever running in a multi-byte locale (so also in UTF-8), hence fixing a regression for users of systems where R 4.1 used a single-byte locale. This was one of the bugs already present in GraphApp, but never reported before, e.g., by users of other multi-byte locales.
- Text injection from external applications via `SendInput` now works in GraphApp Unicode windows, fixing a regression in R 4.2.0 for Rgui users of systems where R 4.1 used a single-byte locale but R 4.2.0 uses UTF-8. Text injection is used via applications such as Dasher, which helps people with hand impairment to enter text, and by general GUIs that prefer to use a standalone Rgui window over embedding R. This was another old bug in GraphApp impacting multi-byte locales. Some other applications injecting text via sending Windows window messages such as `WM_CHAR` directly to Rgui should switch to injection via `SendInput`, which is the proper injection method on Windows and the switch should not be difficult. Allowing injection via `WM_CHAR` even in a multi-byte locale would require too big changes in GraphApp.
- A regression in writing to the clipboard connection has been fixed. That code had to be rewritten for the UTF-8 transition, but the new version had a bug which prevented writing text in consecutive operations.
- `getLocale` has been fixed to also work with strict checking of invalid arguments to the C runtime. These are normally disabled in applications built using Rtools, but it impacted embedded use in RStudio with the `rJava` package, causing crashes by default. This issue was related to switching to UCRT, which is stricter in checking the validity of function arguments (it was not directly related to UTF-8 nor the locale).
- The script editor in Rgui has been fixed to work with UTF-8: some operations (such as running a line of code from the editor in the R console) before did not work with non-ASCII characters, with a regression in R 4.2.0 (in earlier versions one could work at least with non-ASCII characters representable in the current locale). These issues are related to at least surprising behavior of the underlying Windows component used for the editor with UTF-8 as the native (ANSI) encoding: one would think that no change would be needed for the transition from a non-UTF-8 native encoding to UTF-8 here. Users of the script editor have to convert their scripts with non-ASCII characters to UTF-8 before reading them in R 4.2.1 or newer (on recent Windows where UTF-8 is used), and they should upgrade to R 4.2.2 when it is available.

Selected fixes in R-patched, to become R 4.2.2:

- `Rterm` support for `Alt+xxx` sequences has been fixed to produce the corresponding character (only) once. This fixes pasting text that includes tilde on Italian keyboard; without the fix, tilde may appear twice, depending on the Windows console program used. This was a regression introduced by an earlier rewrite of `Rterm` for supporting multi-width characters. That change was part of the transition to UTF-8, but already included in R 4.1.
- Find and replace operations work again in the script editor in Rgui.

The bug reports have revealed that Rgui is frequently used, including by users with visual or hand impairments who find Rgui working well with assistive technologies.

The bugs were fixed promptly in R-patched. Impacted users may install a snapshot of R-patched in case waiting for the next release would be too limiting.

3 R 4.2.0 code statistics

From the source code Subversion repository, the overall change between May 18, 2021 and April 22, 2022 (so between R 4.1.0 and R 4.2.0) was: 29,000 added lines, 20,000 deleted lines and 900 changed files. This is rounded to thousands/hundreds and excludes changes to common generated files, bulk re-organizations, etc. (translations, parsers, Autoconf, LAPACK, R Journal bibliography, test outputs, Unicode tables, incorporated M4 macros, BLAS, KaTeX files). This is about 10% fewer additions and changed files and about 43% more deletions than between R 4.0.0 and R 4.1.0, see News and Notes from the June 2021 issue of the R Journal.

Figure 1 shows commits by month and weekday, respectively, counting line-based changes in individual commits, excluding the files as above. The statistics are computed the same way as in the June 2021 issue, hence allowing direct comparisons. However, monthly statistics are impacted by the release date which varies across versions, so the numbers for April and May are somewhat biased. The statistics cover code directly committed to the trunk, plus commits from the merged branches *R-groups*, *R-vecpat*, *R-ucrt* and *R-structure*. Statistics are based on the dates of the original commits in the branches, which includes commits from April 2021 (*R-groups*).

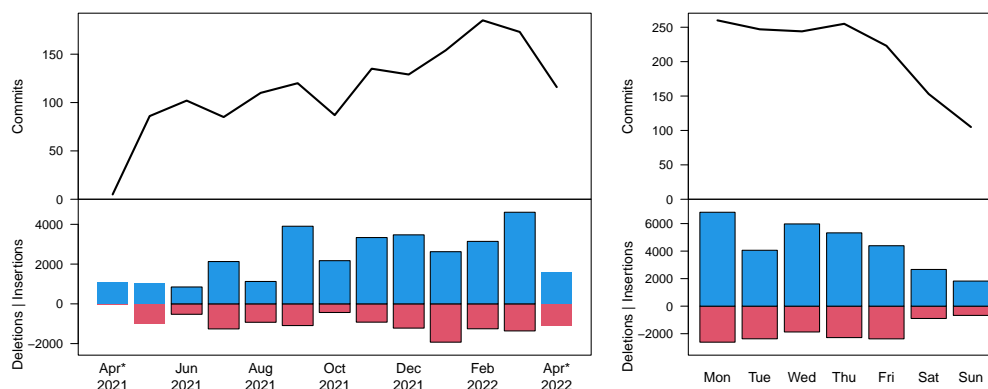


Figure 1: Commit statistics by month (left) and weekday (right) during R 4.2.0 development. *Counts for April 2021 represent early work on the *R-groups* branch. May 2021 and April 2022 are partial months impacted by the release dates.

4 Acknowledgements

Tomas Kalibera's work on the article and R development has received funding from the Czech Ministry of Education, Youth and Sports from the Czech Operational Programme Research, Development, and Education, under grant agreement No.CZ.02.1.01/0.0/0.0/15_003/0000421, from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, under grant agreement No. 695412, and from the National Science Foundation award 1925644.

Tomas Kalibera
Czech Technical University, Czech Republic
Tomas.Kalibera@R-project.org

Sebastian Meyer
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
Sebastian.Meyer@R-project.org

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

R Foundation News

by *Torsten Hothorn*

1 Donations and members

Membership fees and donations received between 2022-03-31 and 2022-09-05.

Donations

New Elements GmbH (Germany) William Chiu (United States) Giles Dickenson-Jones (Australia) Jonathan Keane (United States) Daniel Wollschläger (Germany)

Supporting institutions

Alfred Mueller Analytic Services, München (Germany) Chicago R User Group, Chicago (United States) Ef-prime, Inc., Chuo-ku (Japan) KIN Services, Tijuana (Mexico) University of Iowa, Iowa City (United States)

Supporting members

Vedo Alagic (Austria) Mohammed Almozini (Saudi Arabia) Kristoffer Winther Balling (Denmark) Joaquín Baquer-Miravete (Spain) Ashanka Beligaswatte (Australia) Chris Billingham (United Kingdom) Wesley Brooks (United States) Robert Carnell (United States) Rafael Costa (Brazil) Charles Cowens (United States) Alistair Cullum (United States) Ajit de Silva (United States) Dubravko Dolic (Germany) Mitch Eppley (United States) Guenter Faes (Germany) Leonardo Ferreira (Germany) Gottfried Fischer (Austria) Ainota Galadriota (United Kingdom) Jutta Gampe (Germany) James Harris (United States) Takehiko Hayashi (Japan) Alessandro Ielpi (Canada) ken ikeda (Japan) Anup Jaltade (United States) Knut Helge Jensen (Norway) Brian Johnson (United States) Christian Kampichler (Netherlands) Sebastian Koehler (Germany) Sebastian Krantz (Germany) Luca La Rocca (Italy) Teemu Daniel Laajala (Finland) Jindra Lacko (Czechia) Seungdoe Lee (Korea, Republic of) Zhiguang Li (United States) Eric Lim (United Kingdom) Sharon Machlis (United States) Michal Majka (Austria) harvey minnigh (Puerto Rico) Maciej Nasinski (Poland) Aliaksandr Nekrashevich (Canada) Mark Niemann-Ross (United States) Boris Ntwoku (United States) Jens Oehlschlägel (Germany) Jaesung James Park (Korea, Republic of) Bill Pikounis (United States) Kelly Pisane (Netherlands) Andrzej Pokładek (Poland) Davor Pranjić (United States) Srivatsan Raghunathan (India) Sindri Shtepani (Canada) Murray Sondergard (Canada) Marco Steenbergen (Switzerland) Berthold Stegemann (Germany) Michael Tiefelsdorf (United States) Nicholas Turner (United States) Philipp Upravitelev (Russian Federation) Mark van der Loo (Netherlands) Frans van Dunné (Costa Rica) Jason Wyse (Ireland) Jaejoong Yun (Korea, Republic of) 杨(Yang) 胡(Hu) (New Zealand)

Torsten Hothorn

Universität Zürich, Switzerland Torsten.Hothorn@R-project.org

The Conference Report of Why R? Turkey 2022: The First R Conference with Call For Papers in Turkey

by Mustafa Cavus, Olgun Aydin, Ozan Evkaya, Derya Turfan, Filiz Karadag, Ozancan Ozdemir, Ugur Dar, and Deniz Bezer

Abstract Why R? Turkey 2022 was a non-profit conference that aimed to bring Turkish R users together and encourage them to attend the R conferences. The targeted audience of the conference consisted of, data scientists, data analysts, and all R users from academia and industry. The three-day conference, which consisted of several events such as workshops, regular talks, lightning talks, short tutorials, and panels, was free of charge and fully online. This article describes the challenges and benefits, as well as providing an overview of the conference's content and participants' profile.

1 Motivation

The Why R? Turkey 2022 conference was a three-day online conference that was organized as a part of the Why R? Global conference series have been organized annually by the Why R? Foundation since 2017. It is known as one of the largest annual R conferences in Europe (Burdukiewicz et al. 2018, 2019). In addition to the main conference, pre-meetings are held in different cities from all over the world. One of these pre-meetings was organized with the invited speakers in Turkey in 2020. This online event was the first R conference in Turkey. In addition, as a result of the positive feedback received from the participants, the theme of the Why R? Turkey 2021 conference was determined immediately. Thereafter, the second R conference was organized in 2021 with invited speakers (Cavus et al. 2021). The main objective was *to bring Turkish R users together from all over the world*, the second conference attracted a lot of attention with over 2000 registered participants.

The main limitation of the conference in 2021 was that it consisted of invited speakers only. Upon the feedback from the participants, the conference structure was changed into having regular talks, lightning talks. In addition, the conference also had a practical side with workshops for both beginners and more experienced R users. In this way, the conference turned into a more participant-friendly format. The following subsections are summarizing the planning phase and the content of the events.

2 Planning

After comparing several alternatives, it was decided to continue with *Zoom* (zoom.us) with one main account and six admins as the main platform for all events.

A mailing tool called *MailChimp*, which also has an admin account, was used to establish seamless communication with the scientific committee and all other participants in the organization.

Participation certificates for attendees, speakers, workshop tutors and panelists were prepared



Figure 1: The logo of Why R? Turkey 2022 Conference

and delivered by using the *Sertifier* platform (sertifier.com/tr/) at the end of the conference.

Numerous academic publishing houses were investigated for publishing proceeding book with E-ISBN and it was decided to work with *Nobel Publication House*. The electronic version of the proceeding book was provided by the international publishing house.

Workshops

Five workshops (~ 1.5 hour-long sessions), from beginner level to advanced level were organized. Three of the workshops were held in Turkish and two in English. Thus, it was ensured that participants from all over the world would benefit from them. Detailed contents provided by the tutors for the workshops. Each workshop was recorded with the consent of the tutor and after the conference videos were published on Youtube. Details and Youtube links for each workshop can be found below.

1. **Introduction to R Programming** by **Ahmet Uraz Akgül** (Turkish): R programming language is increasing its popularity day by day, including our country, and it has found use in almost every discipline. Being a free software, it has started to reach more people with its access to open source codes and the opportunities it offers. In such an environment, knowing the R programming language provides serious advantages. In the competitive software world, R can survive as a language worth learning; it may even be the language we use most often.
2. **Data Visualization with R** by **İnan Utku Türkmen** and **Cansu Hürses** (Turkish): As part of the data visualization workshop with R, we will cover basic visualization issues with the **ggplot2** (Wickham 2016) package, which is one of the most popular data visualization tools in the field of data science. Using this package we will practice how to create basic static chart types. We will see with examples how a chart can be customized by adding a title, specifying axis names, coloring according to certain categories, marking a specific region on the chart, and taking a section. During the workshop, we will create the interactive graphic in the link below, which is one of the famous visualizations of the *Gapminder* foundation, using R packages step by step: <https://www.gapminder.org/tag/map/>.
3. **Processing tabular data with tidyverse** by **İmran Kocabıyık** (Turkish): Most of the data we use in practice are kept in tabular format. The purpose of use of these data, which are stored in databases and various types of files, may not always be analysis. For this reason, even if the data is kept in tabular form, it requires serious processing to model or analyze it. The **dplyr** (Wickham et al. 2022) and **tidyr** (Wickham and Girlich 2022) packages offer very simple and effective solutions for these processes.
4. **Introduction to Bioconductor** by **Nitesh Turaga** (English): This talk is an overview of the Bioconductor project which develops, supports, and disseminates free open source software, in the R programming language, that facilitates rigorous and reproducible analysis of data from current and emerging biological assays. The Bioconductor project has a large footprint around the world both in industry and academic research. We'll discuss the important aspects of project such as methods of contribution, programming paradigms - interoperability and data structure design, and differences to other projects to draw comparison. It will also include a brief introduction to essential genomics data structures that have been pillars to the Bioconductor ecosystem. We welcome participation as we are dedicated to building a diverse, collaborative, and welcoming community of developers and data scientists.
5. **Deep Learning using R** by **Krystian Zielinski** (English): Deep Learning is certainly not a future of technologies that surround us on a daily basis. It's too late – it's already here. Just try to find a company that's not willing to enhance its products with AI driven algorithms. The data is there already, and the trust barrier is shrinking. It's to nobody's surprise – if AI is used in very sensitive areas like banking, military or medicine, it can't be that bad. Netflix's recommendation algorithms, Google Lens, Self Driving cars – you name it, Deep Learning is used everywhere. Thanks to many popular frameworks, training Neural Network model has never been so easy. You can find tutorials online, and based on them solve many problems. But after some time, the question arises: „How does this even work?“ – and you should really know the answer. In the workshops I'd like to show you how theory affects the practice. Based on many examples we will train DNN models, show its limits, check how parameters like e.g. activation function affect the training phase, suggest best practices in data preparation and DNN architecture, explain the model's predictions. If you're familiar with R – this workshop is suited for you! If you're just starting with Deep Learning or already have some experience, come and join us - I bet you won't regret it!

Short Tutorials

In addition to the workshops, short tutorials on three topics were organized. This type of event was planned to focus on more specific topics in longer duration than a regular talk. (~ 30 mins each) Each workshop was recorded with the consent of the tutor and after the conference videos were published on Youtube. Details and Youtube links for each workshop can be found below.

1. [Spatial Analysis with R](#) by [Fırat Gündem](#) (Turkish): Open Data Portal applications of local governments both in America and Europe added a serious spatial dimension to data. The spatial dimension in question covers a wide area, from storing data in a way that includes spatial information (shape file, GeoJSON, etc.) to spatial data visualizations and analysis with spatial statistics and spatial econometric methods. The R program is constantly being renewed and expanded to include all spatial analysis techniques in the literature. Although there are many people who do spatial analysis with package programs such as ArcGIS, which are expensive and require personal licenses, RStudio allows to perform all spatial analyzes easily and free of charge with the libraries it contains. In this short study, all the basic steps of spatial analysis will be performed using a set of R libraries. For this, real spatial data from Turkey (GDP per capita on a provincial basis, etc.) will be used. Then, spatial data projection (`sf`, `sp`), neighborhood matrix creation and manipulation (`rgdal`), spatial data visualization and static and dynamic mapping (`ggplot2` (Wickham 2016), `tmap` (Tennekes 2018), `leaflet` (Cheng, Karambelkar, and Xie 2021), three-dimensional mapping `rayshader` (Morgan-Wall 2021), spatial statistics `rgeoda` (Li and Anselin 2022) in RStudio. and spatial econometric models will be introduced practically using the relevant R libraries. Thus, all necessary tools for spatial data science will be introduced in R and the use of R will be encouraged with user-friendly applications.
2. The R Application for Physics-Informed Neural Networks by [Melih Ağraz](#) (Turkish): Physics-Informed Neural Networks (PINNs) is a deep learning framework designed to solve nonlinear differential equations using artificial neural networks, published in 2019. Simple deep feed-forward neural network architectures and automatic differentiation method are used to solve differential equations in PINNs method. The PINNs method was first developed with Python Tensorflow. In this study, we will show how the solution of the equation $y' - y = 0$ for $y(0) = 1$, $y(1) = e$ is solved by using `reticulate` (Ushey, Allaire, and Tang 2021) library in R, with the help of PINNs method. This study is thought to find application area of PINNs method for R users as well. For this reason, a simple differential equation solution example such as $y' - y = 0$ is preferred.
3. [Serverless R in the Cloud - Deploying R into Production with AWS and Docker](#) by [İsmail Tigrek](#) (English): This tutorial will walk through deploying R code, machine learning models, or Shiny applications in the cloud environment. With this knowledge, you will be able to take any local R-based project you've built on your machine or at your company and deploy it into production on AWS using modern serverless and microservices architectures. In order to do this, you will learn how to properly containerize R code using Docker, allowing you to create reproducible environments. You will also learn how to set up event-based and time-based triggers. We will build out a real example that reads in live data, processes it, and writes it into a data lake, all in the cloud.

Panels

The panels were planned based on the sharing experience in an interactive environment, where the participants could come together with experts in different domains. *Biostatistics* and *R Education* were selected as major topics in these events. The links to the video recordings and detailed content of the panels are given below:

1. [Biostatistics and Applications in R](#) moderated by Prof. Dr. Ergun Karaağaoğlu (Turkish): Prof. Dr. Recai Yücel (Temple University), Prof. Dr. Mithat Gönen (Memorial Sloan Kettering Cancer Center), and Dr. Anil Dolgun (CSL Limited) attended as the panelists. In this panel, current research studies on biostatistics and the importance of using R in this domain were discussed. In addition, information about how career development was positively affected by using R was shared by the panelists with the audience. Thus, it was ensured that the participants were informed about the importance of the R in this field.
2. [R Education](#) moderated by [Olgun Aydın](#) (Turkish): The main topic of the panel was how to teach R in academia. Prof. Dr. Mine Çetinkaya-Rundel (Duke University), Dr. Mine Doğucu (University of California), and Assoc. Prof. Dr. Kübra Kabasakal (Hacettepe University) attended as the panelists. The panelists shared their approach in terms of teaching R in different fields of academic studies. Moreover, the panelists shared examples from their teaching materials with the audience.

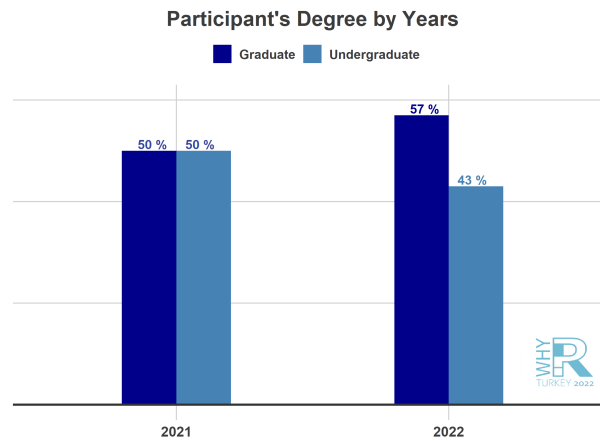


Figure 2: The percentage of the degree of the participant students by years

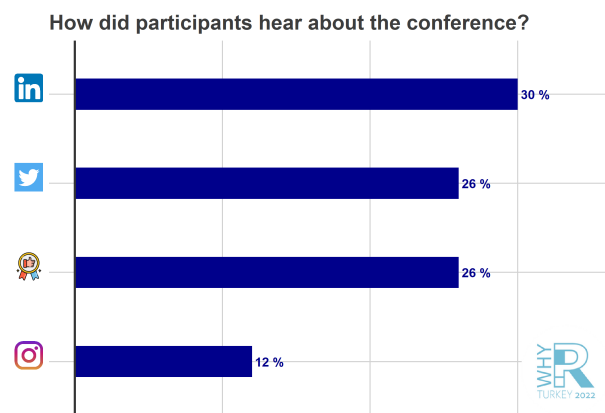


Figure 3: How did the participants hear about the conference?

3 Participants

The 1366 participants were registered for the conference: 70% of the participants were students and the rest were professionals who work for governmental institutions, companies from the private sector, and as academicians. The 43% of the students are undergraduates, 26% of them master studies and the 31% are Ph.D. students.

When the channels that the participants heard about the conference are examined, surprisingly, the percentage of participants who registered for the conference "on recommendation" was quite high (26%), in addition to the social media channels (LinkedIn - 30%, Twitter - 26%, and Instagram - 12%) of which usage has increased in recent years.

In the registration form, questions about participants' level of R knowledge and motivation of participants' to take part in Why R? Turkey 2022 were included to deeply analyze profile of the participants. The 69% of the participants stated that they had taken or attended a course related to R before. They evaluated their R programming language usage levels as shown in Table 1.

It can be seen from Table 1 that 90% of them did not attend a conference about R before. In this respect, the conference was held with an audience suitable for the purpose of expanding usage of R.

Table 1: The R Programming language usage levels of the participants

Level	Percent
Beginner	38
Intermediate	31
Advanced	9
Never used	22

The areas of their interest are *Data Visualization*, *Statistical Modeling*, *Big Data*, *Data Mining*, *Machine Learning*, and *Deep Learning*, respectively. The events planned for Why R? Turkey 2022 that motivated them register the conference are respectively: *Scientific Sessions* (40%), *Workshops* (35%) and *Panels* (26%).

4 Evaluation

An evaluation questionnaire was conducted after the event to measure the satisfaction with the conference and to collect suggestions future organizations (Cordoba et al. 2019). The satisfaction part of the questionnaire consists of two questions: (1) general satisfaction and (2) the interaction level between the presenters and the listeners. The results of the questionnaire are quite positive in terms of the answer of the respondents. On a scale of 0 – 5, the mean of the satisfaction score is 4.74 with 0.46 standard deviation. Besides that, the mean of the interaction score is 4.50 with 0.69 standard deviation.

The second part of the questionnaire consists of the questions related to the motivation of the respondents regarding attendance and being a presenter for the next edition of the conference. According to the responses, although nearly all of the respondents (99%) plan to attend the conference in the next years, 30% aim to participate to as a presenter.

The last part of the questionnaire is about the suggestions from the respondents. The suggestions generally focus on increasing the duration of regular talks and workshops. In addition, among the speakers, the higher number of private sector employees and the mention of sector or freelance job opportunities were among the most prominent suggestions. Respondents strongly mentioned that next events, which has been held online for two years, should be held face-to-face in the coming years to increase its effectiveness.

5 Summary

Why R? Turkey 2022, as the next edition of Why R? Turkey 2021 conference, was the first R conference with call for papers held in Turkey. This conference aimed to encourage researchers and professionals, who are R users and developers, to share their work with a wider audience. This conference was organized as a user-friendly conference, taking into account the feedback received in the previous year's version. In addition to scientific sessions, workshops, short tutorials, and panels were organized to broaden the content of the event. According to the evaluation survey made right after the conference, the participants stated that they were very satisfied and they would like to participate as a presenter to Why R conferences in the upcoming years. This finding demonstrates that the organisers of the conference succeeded the goals.

6 Organizers

The Organizing Committee of Why R? Turkey 2022 consisted of eight Turkish researchers from three different countries: Turkey, Poland, and United Kingdom. It consisted of [Mustafa Cavus](#) (Warsaw University of Technology, Eskisehir Technical University), [Olgun Aydin](#) (Gdansk University of Technology, Why R? Foundation), [Filiz Karadag](#) (Ege University), [Ugur Dar](#) (Eskisehir Technical University), [Ozan Evkaya](#) (University of Edinburgh), [Derya Turfan](#) (Hacettepe University), [Ozancan Ozdemir](#), [Deniz Bezer](#) (Middle East Technical University).

7 Acknowledgement

As an organization committee, we would like to thank [Marcin Kosinski](#) for his suggestions and support during the preparation part of the conference, [Klaudia Korniluk](#) for designing our graphics, and [Erdal Cosgun](#), [Gokmen Zararsiz](#) for their guidance in general and also organizing the some of the panels and workshops. Moreover, we would like to thank the sponsors that support the Why R? Turkey 2022, [R Studio](#), [Visbanking](#) as gold sponsor, [CRC Press](#) as the award sponsor, [Appsilon](#) and [Jumping Rivers](#) silver sponsor.

8 Additional Information

Further information about the conference such as website of the conference, recordings for regular, lightning talks, workshops and panels, materials, proceeding book and social media accounts used for the conference can be reached out via following links:

- Website: <https://whyr.pl/2022/turkey/>
- Abstract book: <https://www.nobelyayin.com/why-r-turkiye-2022-konferansi-18447.html>
- YouTube channel: <https://www.youtube.com/c/WhyRTurkey>
- Twitter: <https://twitter.com/whyrturkey>
- LinkedIn: <https://www.linkedin.com/company/why-r-turkey/>
- Instagram: <https://www.instagram.com/whyrturkey/>

References

- Burdukiewicz, M., M. Karas, L. E. Jessen, M. Kosinski, B. Bischl, and S. Rödiger. 2018. "Conference Report: Why r? 2018." *R Journal* 10 (2): 572–78. <https://doi.org/10.32614/RJ-2018-2-whyR>.
- Burdukiewicz, M., F. Pietluch, J. Chilimoniuk, K. Sidorczuk, D. Rafacz, L. E. Jessen, S. Rödiger, M. Kosinski, and P. Wojcik. 2019. "Conference Report: Why r? 2019." *R Journal* 12 (1): 484–93. <https://doi.org/10.1080/10618600.1996.10474713>.
- Cavus, M., O. Aydin, O. Evkaya, O. Ozdemir, D. Bezer, and U. Dar. 2021. "Conference Report of Why r? Turkey 2021." *R Journal* 13 (1): 648–53. <https://journal.r-project.org/archive/2021-1/whyr2021.pdf>.
- Cheng, Joe, Bhaskar Karambelkar, and Yihui Xie. 2021. *Leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*. <https://CRAN.R-project.org/package=leaflet>.
- Cordoba, M. A., F. Dunne, A. G. Melendez, and J. Eten. 2019. "Conference Report: ConectaR2019." *R Journal* 11 (2): 439–42.
- Li, Xun, and Luc Anselin. 2022. *Rgeoda: R Library for Spatial Data Analysis*.
- Morgan-Wall, Tyler. 2021. *Rayshader: Create Maps and Visualize Data in 2d and 3d*. <https://CRAN.R-project.org/package=rayshader>.
- Tennekes, Martijn. 2018. "tmap: Thematic Maps in R." *Journal of Statistical Software* 84 (6): 1–39. <https://doi.org/10.18637/jss.v084.i06>.
- Ushey, Kevin, JJ Allaire, and Yuan Tang. 2021. *Reticulate: Interface to 'Python'*. <https://CRAN.R-project.org/package=reticulate>.
- Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.
- Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2022. *Dplyr: A Grammar of Data Manipulation*.
- Wickham, Hadley, and Maximilian Girlich. 2022. *Tidyr: Tidy Messy Data*.

Mustafa Cavus
Eskisehir Technical University
Department of Statistics
Eskisehir, Turkey
ORCID: 0000-0002-6172-5449
mustafacavus@eskisehir.edu.tr

Olgun Aydin
Gdansk University of Technology
Gdansk, Poland
ORCID: 0000-0002-7090-0931
olgun.aydin@pg.edu.pl

Ozan Evkaya
The University of Edinburgh
School of Mathematics
Edinburgh, UK
ORCID: 0000-0002-5076-8144
ozanevkaya@gmail.com

Derya Turfan
Hacettepe University

Department of Statistics
Ankara, Turkey
ORCID: 0000-0001-8252-1325
deryaturfan@hacettepe.edu.tr

Filiz Karadag
Ege University
Department of Statistics
Izmir, Turkey
ORCID: 0000-0002-0116-7772
filiz.karadag@ege.edu.tr

Ozancan Ozdemir
Middle East Technical University
Department of Statistics
Ankara, Turkey
ORCID: 0000-0002-7850-3885
ozancan@metu.edu.tr

Ugur Dar
Eskisehir Technical University
Department of Statistics
Eskisehir, Turkey
ugurdar@eskisehir.edu.tr

Deniz Bezer
Middle East Technical University
Department of Statistics
Ankara, Turkey
deniz.bezer@metu.edu.tr