The R Journal

Volume 7/1, June 2015

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

Contents

Editorial

by Bettina Grün

On behalf of the editorial board, I am pleased to publish Volume 7, Issue 1 of the R Journal. This issue contains 16 contributed research articles. Each of them either presents an R package, a specific extension of an R package or applications using R packages available from the Comprehensive R Archive Network (CRAN, `http://CRAN.R-project.org`). It thus provides a small insight into the wide variety of functionality covered currently by the more than 6800 packages available from CRAN.

The presented packages include packages for enhancing the graphics functionality of R such as package **gridGraphics** for converting graphics drawn with the **graphics** package to **grid** graphics and **showtext** for using system fonts in R graphics. Additional graphical tools are provided by package **sparkTable**, which allows to enhance tables, and by package **fanplot**, which allows to visualize the uncertainty connected with forecasts using fan charts. Further infrastructure is implemented in package **rstackdeque** which provides efficient data structures for stacks and queues.

Some of the presented packages provide specialized infrastructure which is valuable for certain areas of application or data situations such as the **Peptides** package for antimicrobial peptides analysis, the **Frames2** package for estimation in dual frame surveys, packages **dpcr** and **qpcr** for the analysis of data from digital and quantitative polymerase chain reaction experiments and package **fslr** which provides a connection to the FSLR software commonly used to process and analyze neuroimaging data.

Specific statistical methods and models which might prove useful in different areas of applications are provided by package **rdrobust**, which allows for robust nonparametric inference in regression-discontinuity designs, by package **cmvnorm**, which implements a complex generalization of the **mvtnorm** package, by package **sae**, which complements other software available on CRAN for small area estimation, by package **FactoMineR**, which now also contains functionality for correspondence analysis on generalized aggregated lexical tables, by package **cna**, which performs coincidence analysis to identify complex causal dependencies, and package **estimability**, which allows to determine if a certain prediction is possible in a rank-deficient regression. Furthermore package **discreteRV** provides infrastructure to manipulate discrete random variables which is intended to help students in introductionary probability courses to understand the theoretical concepts and thus adds to the other tools available in R for teaching.

In addition the News and Notes section contains the usual updates on the R Foundation, the Bioconductor project, CRAN, and changes in R itself.

I hope you enjoy the issue.

Bettina Grün
`Bettina.Gruen@jku.at`

Peptides: A Package for Data Mining of Antimicrobial Peptides

by Daniel Osorio, Paola Rondón-Villarreal and Rodrigo Torres

Abstract Antimicrobial peptides (AMP) are a promising source of antibiotics with a broad spectrum activity against bacteria and low incidence of developing resistance. The mechanism by which an AMP executes its function depends on a set of computable physicochemical properties from the amino acid sequence. The **Peptides** package was designed to allow the quick and easy computation of ten structural characteristics own of the antimicrobial peptides, with the aim of generating data to increase the accuracy in classification and design of new amino acid sequences. Moreover, the options to read and plot XVG output files from GROMACS molecular dynamics package are included.

Introduction

Antimicrobial peptides are a promising source of antibiotics with a broad spectrum activity against bacteria and low incidence of developing resistance (Hancock, 2001). Multiple research has concluded that the natural biological activities of these peptides are coordinated by a sophisticated modulation of the hydrophobicity, amphipathicity, positive charge and a reduction in the hydrophobic moment (Yeaman and Yount, 2003; Fjell et al., 2012; Matsuzaki, 2009). Additionally to these four properties there are other descriptors that used in conjunction can provide useful information in the classification and design of antimicrobial peptides (Boman, 2003; Wang et al., 2009; Thomas et al., 2010; Piotto et al., 2012).

The computation of these properties for antimicrobial peptides is available free of charge through various web or native applications as ExPASy-protparam (Gasteiger et al., 2005), EMBOSS-pepstats (Rice et al., 2000), BioPerl (Stajich et al., 2002), CAMP (Thomas et al., 2010) and APD (Wang et al., 2009) databases. However, no application allows the computation of all properties. Some of these services allow only the computation of some properties for a sequence at a time, and have no option for downloading and handling this information in editable files. Others allow the calculation of more than one sequence, but they write an output file for each one of them, which makes it difficult to handle and analyse the data.

Taking the advantage of handling data, vectors and tables provided by R, we introduce the **Peptides** package. It allows quick and easy computation of ten structural characteristics own of the antimicrobial peptides (length, amino-acid composition, net charge, aliphatic index, molecular weight, isoelectric point, hydrophobic moment, potential peptide interaction index, instability index and GRAVY hydrophobicity index) in a single application. The **Peptides** package was designed with the aim of generating data to increase the accuracy in the classification process of new amino acid sequences. In addition, the option to read and plot XVG output files from GROMACS molecular dynamics package was included.

In this work we describe the computation of structural properties of the Lasiocepsin peptide (GLPRKILCAIAKKKGKCKGPLKLVCKC) using the **Peptides** package, step by step as an example. It is an alpha-helical antimicrobial peptide derived from the venom of eusocial bee *Lasioglossum laticeps*, identified in the Protein Data Bank with the 2MBD code (Monincová et al., 2012). Moreover, an example of a classification using a data set of 23 variables computed for 100 peptides through **Peptides** is performed. It was found that using this data set for classification through linear discriminant analysis and classification-regression trees allows to classify antimicrobial peptides with an accuracy of 95% and 85%, respectively.

Installation and functions

Peptides includes thirteen functions and is available for download and installation from CRAN, the Comprehensive R Archive Network. To install it, just type:

```
> install.packages("Peptides")
> library(Peptides)
```

The **Peptides** package requires R version 1.2.2 or higher. Development releases of the package are available on the GitHub repository `http://github.com/dosorio/peptides`.

Number of amino acids

As all proteins, the antimicrobial peptides are formed by linear chains of small residues known as amino acids attached to each other by peptide bonds. Antimicrobial peptides are characterized by a short length, they generally comprise less than 50 amino acids. This property minimizes the probability of being degraded by bacterial proteases (Kim et al., 2013). The function lengthpep counts the number of amino acids in a sequence and returns a vector with the count for each peptide used as argument.

```
> lengthpep(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC")
```

```
[1] 27
```

Molecular weight

The molecular weight is the sum of the masses of each atom constituting a molecule. The molecular weight is directly related to the length of the amino acid sequence and is expressed in units called daltons (Da). Antimicrobial peptides due to its short length are characterized by a molecular weight <10 kDa (10000 Da). In **Peptides** the function mw computes the molecular weight using the same formulas and weights as ExPASy's "compute pI/mw" tool (Gasteiger et al., 2005).

```
> mw(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC")
```

```
[1] 2897.787
```

Amino acid composition

Amino acids are zwitterionic molecules with an amine and a carboxyl group present in their structure. Some amino acids possess side chains with specific properties that allow grouping them in different ways. The aacomp function classifies amino acids based on their size, side chains, hydrophobicity, charge and their response to pH 7 following the categories listed in Table 1. The output is a matrix with the number and percentage of amino acids of a particular class.

Class	Amino acids
Tiny	A C G S T
Small	A B C D G N P S T V
Aliphatic	A I L V
Aromatic	F H W Y
Non-polar	A C F G I L M P V W Y
Polar	D E H K N Q R S T Z
Charged	B D E H K R Z
Basic	H K R
Acidic	B D E Z

Table 1: Classification of the amino acids according to the properties of size and side chain.

Antimicrobial peptides are amphipathic (with similar proportions of polar and non-polar amino acids) and charged molecules. In **Peptides** the amino acid composition can be computed using the function aacomp.

```
> aacomp(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC")
```

```
          Number  Mole%
Tiny           9 33.333
Small         12 44.444
Aliphatic      9 33.333
Aromatic       0  0.000
NonPolar      18 66.667
Polar          9 33.333
Charged        9 33.333
Basic          9 33.333
Acidic         0  0.000
```

Net charge

Some side chains of certain amino acids can confer electric charge to the proteins under certain pH values. The sum of the charges of each of the amino acids is called net charge. Antimicrobial peptides have a positive net charge (of at least +2) at pH 7, which provides binding specificity to the negatively charged bacterial membranes through electrostatic interactions (Yeaman and Yount, 2003).

The `charge` function compute the net charge using Equation 1, a variation of the Henderson Hasselbalch equation proposed by Moore (1985) wherein N are the number, j and i index represent the acidic (Aspartic Acid, Glutamic Acid, Cysteine and Tyrosine) and basic (Arginine, Lysine, and Histidine) functional groups of amino acids, respectively.

The net charge of a protein can be calculated specifying the pH value and one of the nine pKa scales availables (Bjellqvist, Dawson, EMBOSS, Lehninger, Murray, Rodwell, Sillero, Solomon or Stryer).

$$charge = \sum_i N_i \frac{+1}{1+10^{+(pH-pK_{ai})}} + \sum_j N_j \frac{-1}{1+10^{-(pH-pK_{aj})}} \tag{1}$$

```
> charge(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC", pH = 7, pKscale = "EMBOSS")

[1] 8.85201
```

Isoelectric point

The isoelectric point (pI) is the pH at which the net charge of the protein is equal to 0. It is a variable that affects the solubility of the peptides under certain conditions of pH. When the pH of the solvent is equal to the pI of the protein, it tends to precipitate and loose its biological function. Antimicrobial peptides have an isoelectric point close to 10 (Torrent et al., 2011), which is very similar to soap or detergent and consistent with the proposed mechanisms of action for these peptides.

The calculation of the isoelectric point of a peptide may be performed through the function `pI` specifying one of the nine pKa scales available (Bjellqvist, Dawson, EMBOSS, Lehninger, Murray, Rodwell, Sillero, Solomon or Stryer).

```
> pI(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC", pKscale = "EMBOSS")

[1] 10.801
```

Aliphatic index

It has been suggested that the aliphatic amino acids (A, I, L and V) are responsible for the thermal stability of proteins. The aliphatic index was proposed by Ikai (1980) and evaluates the thermostability of proteins based on the percentage of each of the aliphatic amino acids that build up proteins. This index is computed using Equation 2 wherein X_A, X_V, X_I and X_L are the mole percent (100 x mole fraction) of Alanine, Valine, Isoleucine and Leucine respectively.

$$AI = X_A + 2.9X_V + 3.9\,(X_I + X_L) \tag{2}$$

Antimicrobial peptides tend to be more thermostable than proteins in general. For the calculation of the aliphatic index, the function `aindex` was included.

```
> aindex(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC")

[1] 104.8148
```

Instability index

The instability index was proposed by Guruprasad et al. (1990). This index predicts the stability of a protein based on its amino acid composition. It is calculated according to Equation 3 where L is equal to the length of the amino acid sequence; X_iY_i is a dipeptide and $DIWV$ is the dipeptide weight value on amino acid sequence of stable proteins.

$$II = \left(\frac{10}{L}\sum_{i=10}^{L-1} DIWV_{(X_iY_i+1)}\right) \tag{3}$$

Despite their short length (variable that this function penalizes), antimicrobial peptides tend to be considered stable with index values less than 40. The instability index can be calculated using the function `instaindex` incorporated in **Peptides**.

```
> instaindex(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC")
```

```
[1] 2.237037
```

Boman index

The potential protein interaction index was proposed by Boman (2003) as an easy way to differentiate the action mechanism of hormones (protein-protein) and antimicrobial peptides (protein-membrane) through this index. It is calculated using Equation 4 by adding each amino acid solubilities divided by the sequence length. This function predicts the potential peptide interaction with another protein.

$$B_{index} = \frac{\sum_{i=1}^{N} S_i}{N} \tag{4}$$

During its mechanism of action, antimicrobial peptides tend to not interact with other proteins (the proposed mechanism of action is based on the interaction with membranes), so the values for the Boman index are usually negative or nearby to 0. To calculate the Boman index, the boman function is included within **Peptides**.

```
> boman(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC")
```

```
[1] 0.5259259
```

Hydrophobicity index

The hydrophobicity is an important stabilization force in protein folding; this force changes depending on the solvent in which the protein is found. It is considered the driving force of the peptide to the core of the membrane. The hydrophobicity index is calculated following the Equation 5, adding the hydrophobicity of individual amino acids and dividing this value by the length of the sequence. Highly expected transmembrane peptides generally have higher hydrophobicity values than 0.5 using Eisenberg scale.

$$H = \frac{\sum_{i=1}^{N} H_i}{N} \tag{5}$$

Peptides includes thirty-eight scales of hydrophobicity (Aboderin, AbrahamLeo, Argos, BlackMould, BullBreese, Casari, Chothia, Cid, Cowan3.4, Cowan7.5, Eisenberg, Engelman, Fasman, Fauchere, Goldsack, Guy, HoppWoods, Janin, Jones, Juretic, Kidera, Kuhn, KyteDoolittle, Levitt, Manavalan, Miyazawa, Parker, Ponnuswamy, Prabhakaran, Rao, Rose, Roseman, Sweet, Tanford, Welling, Wilson, Wolfenden or Zimmerman) obtained from various sources (Gasteiger et al., 2005; Kawashima et al., 2008).

```
> hydrophobicity(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC", scale = "Eisenberg")
```

```
[1] -0.08777778
```

Hydrophobic moment index

The hydrophobic moment was proposed by Eisenberg et al. (1982), as a quantitative measure of the amphiphilicity perpendicular to the axis of any periodic peptide structure. To calculate the hydrophobic moment, the function hmoment was included. The hydrophobic moment is computed according to Equation 6, using the standardized Eisenberg (1984) scale, windows (fragment of sequence) of eleven amino acids and specifying the rotational angle at which it should be calculated. Highly expected transmembrane peptides generally have lower hydrophobic moment values than 0.2.

$$\mu H = \left\{ \left[\sum_{n=1}^{N} H_n \sin(\delta n) \right]^2 + \left[\sum_{n=1}^{N} H_n \cos(\delta n) \right]^2 \right\}^{\frac{1}{2}} \tag{6}$$

```
> hmoment(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC", angle = 100, window = 11)
```

```
[1] 0.6170697
```

```
> hmoment(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC", angle = 160, window = 11)
```

```
[1] 0.4617153
```

Membrane position

Eisenberg et al. (1982) found a correlation between hydrophobicity and hydrophobic moment that defines the protein section as globular, transmembrane or superficial. The function calculates the hydrophobicity (H) and hydrophobic moment (μH) based on the standardized scale of Eisenberg (1984) using windows of 11 amino acids for calculate the theoretical fragment type.

```
> membpos(seq = "GLPRKILCAIAKKKGKCKGPLKLVCKC", angle = 100)
```

```
           Pep      H    uH  MembPos
1  GLPRKILCAIA  0.271 0.469  Surface
2  LPRKILCAIAK  0.091 0.617  Surface
3  PRKILCAIAKK -0.142 0.520 Globular
4  RKILCAIAKKK -0.289 0.401 Globular
5  KILCAIAKKKG -0.015 0.325 Globular
6  ILCAIAKKKGK -0.015 0.319 Globular
7  LCAIAKKKGKC -0.115 0.339 Globular
8  CAIAKKKGKCK -0.347 0.115 Globular
9  AIAKKKGKCKG -0.330 0.096 Globular
10 IAKKKGKCKGP -0.375 0.141 Globular
11 AKKKGKCKGPL -0.405 0.161 Globular
12 KKKGKCKGPLK -0.597 0.110 Globular
13 KKGKCKGPLKL -0.365 0.156 Globular
14 KGKCKGPLKLV -0.130 0.310 Globular
15 GKCKGPLKLVC  0.033 0.257 Globular
16 KCKGPLKLVCK -0.147 0.426 Globular
17 CKGPLKLVCKC  0.015 0.487 Globular
```

GROMACS files

Molecular dynamics is a source of *in-silico* biophysical data that contribute to the design, classification and testing of antimicrobial peptides. GROMACS (GROningen MAchine for Chemical Simulations) is a molecular dynamics package designed for simulations of proteins, lipids and nucleic acids. It is free, open source software released under the GNU General Public License.

The file format used by GROMACS is XVG. This format can be displayed in graphical form through the GRACE program on UNIX/LINUX systems and the GNUPlot program on Windows. XVG files are plain text files containing tabular data separated by tabulators and two types of comments which contain data labels. Although manual editing is possible, this is not a viable option when working with multiple files of this type.

For ease of reading, information management and data plotting, the functions `read.xvg` and `plot.xvg` were incorporated. An example of how to read and plot the absolute distance between the center of mass of an antimicrobial peptide with respect to a POPG (*1-palmitoyl-2-oleoyl-sn-glycero-3-phosphoglycerol*) pure lipid bilayer is presented below (Figure 1).

```
> file <- system.file(file = file.path("xvg-files", "POPG.xvg"),
+                     package = "Peptides")
> md <- read.xvg(file)
> head(md)
```

```
  Time (ps)      |d|   d\\sx\\N   d\\sy\\N  d\\sz\\N
1         0 3.476546 -0.2250402 -0.3378360 -3.452766
2         3 3.447776 -0.2403412 -0.4272313 -3.412751
3         6 3.459584 -0.2213712 -0.3733103 -3.432252
4         9 3.391920 -0.2328529 -0.3787930 -3.362650
5        12 3.403695 -0.1856968 -0.2425249 -3.389962
```

```
> file <- system.file(file.path("xvg-files", "POPG.xvg"), package = "Peptides")
> plot.xvg(file)
```

Using data in the classification process

Altogether the data produced in **Peptides** are a useful source of specifications allowing classification of antimicrobial peptides and their differentiation from other peptides such as hormones. An example

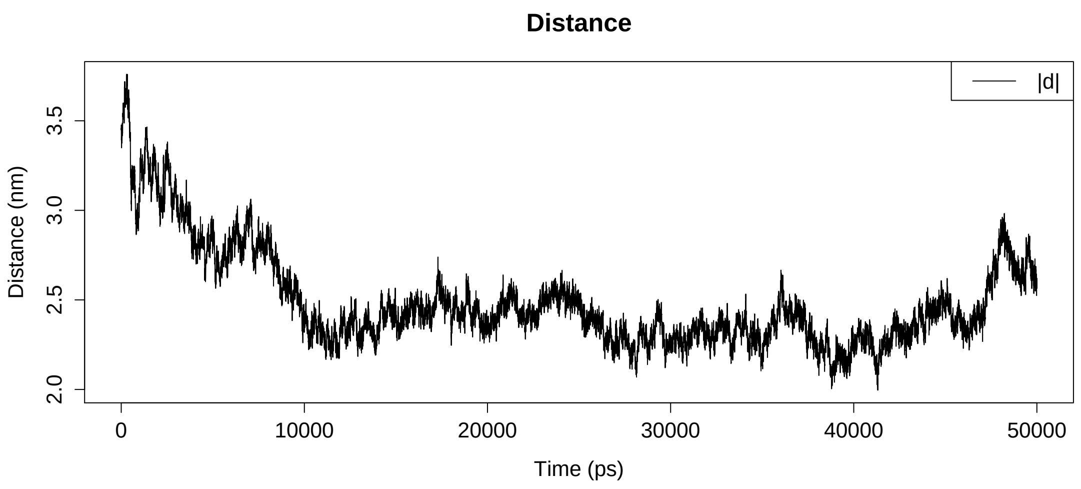


Figure 1: Absolute distance between the center of mass of an antimicrobial peptide with respect to a POPG pure lipid bilayer.

of classification using linear discriminant analysis and classification-regression trees through the **caret** package (Kuhn, 2014) is shown.

```
> install.packages("caret", dependencies = TRUE)
> library(Peptides)
> library(caret)
```

The data set was calculated using all the functions included in **Peptides**. It is available in the `data.frame` with 100 observations and 23 variables called `pepdata`. It includes the physicochemical properties and calculable indices for 50 antimicrobial (group = 1) and 50 non-antimicrobial (group = 0) peptides downloaded from the Protein Data Bank (Bernstein et al., 1978) and the APD database (Wang et al., 2009) respectively.

```
> data(pepdata)
> str(pepdata)
```

```
'data.frame':  100 obs. of  23 variables:
 $ sequence      : chr  "DAEFRHDSGYEVHHQKLVFFAEDVGSNK" "SLDRSSCFTGSLDSIRAQSGLGCNSFRY" ...
 $ group         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ length        : int  28 28 30 22 25 28 30 25 31 26 ...
 $ mw            : num  3263 3028 3275 2747 2903 ...
 $ tinyAA        : num  21.4 50 40 13.6 24 ...
 $ smallAA       : num  46.4 60.7 56.7 27.3 44 ...
 $ aliphaticAA   : num  21.4 17.9 16.7 36.4 28 ...
 $ aromaticAA    : num  25 10.71 3.33 18.18 12 ...
 $ nonpolarAA    : num  42.9 46.4 46.7 54.5 56 ...
 $ polarAA       : num  57.1 53.6 53.3 45.5 44 ...
 $ chargedAA     : num  42.9 17.9 26.7 27.3 28 ...
 $ basicAA       : num  21.4 10.7 20 22.7 16 ...
 $ acidicAA      : num  21.43 7.14 6.67 4.54 12 ...
 $ charge        : num  -2.97 0.742 3.741 1.936 0.762 ...
 $ pI            : num  5.43 8.22 10.05 11.05 9.03 ...
 $ aindex        : num  52.1 59.3 48.7 119.5 89.6 ...
 $ instaindex    : num  26.9 58 45.5 52.8 45.1 ...
 $ boman         : num  2.65 2.43 2.22 1.77 1.84 ...
 $ hydrophobicity: num  -0.9 -0.286 -0.62 -0.145 -0.372 -0.004 -0.23 0.536 0.477 0.754 ...
 $ hmoment       : num  0.392 0.407 0.46 0.533 0.373 0.603 0.483 0.643 0.264 0.38 ...
 $ transmembrane : num  0 0 0 0 0 0 0 0.067 0.095 0.188 ...
 $ globular      : num  1 1 1 1 1 1 0.8 0.733 0.905 0.688 ...
 $ surface       : num  0 0 0 0 0 0 0.2 0.2 0 0.125 ...
```

The training and testing data sets were created with the following commands.

```
> set.seed(2014)
> inTrain <- createDataPartition(y = pepdata$group, p = 0.8, list = FALSE)
> training <- pepdata[ inTrain, 2:23]
> testing <- pepdata[-inTrain, 2:23]
```

To evaluate the potential of these data, the best CART model was selected by a *K*-fold cross-validation loop.

```
> folds <- 10
> repeats <- 10
> fitControl <- trainControl(method = "repeatedcv",
+                            number = folds,
+                            repeats = repeats,
+                            classProbs = TRUE,
+                            allowParallel = TRUE,
+                            summaryFunction = twoClassSummary)
>  train.rpart <- train(group~., data = training,
+                       method = "rpart",
+                       metric  = "ROC",
+                       tuneLength = 10,
+                       trControl = fitControl)
> train.rpart

CART

80 samples
21 predictors
 2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)

Summary of sample sizes: 72, 72, 72, 72, 72, 72, ...

Resampling results across tuning parameters:

  cp          ROC        Sens    Spec    ROC SD     Sens SD    Spec SD
  0.00000000  0.8321875  0.8075  0.7525  0.1327056  0.1875379  0.2117001
  0.07222222  0.7862500  0.7525  0.7700  0.1664132  0.2056494  0.2294041
  0.14444444  0.7462500  0.6450  0.8850  0.1847590  0.2414707  0.2023224
  0.21666667  0.7475000  0.6450  0.9000  0.1854731  0.2414707  0.1946247
  0.28888889  0.7475000  0.6450  0.9000  0.1854731  0.2414707  0.1946247
  0.36111111  0.7475000  0.6450  0.9000  0.1854731  0.2414707  0.1946247
  0.43333333  0.7475000  0.6450  0.9000  0.1854731  0.2414707  0.1946247
  0.50555556  0.7475000  0.6450  0.9000  0.1854731  0.2414707  0.1946247
  0.57777778  0.7475000  0.6450  0.9000  0.1854731  0.2414707  0.1946247
  0.65000000  0.5550000  0.7650  0.3800  0.1128152  0.2926283  0.4570989

ROC was used to select the optimal model using  the largest value.
The final value used for the model was cp = 0.

> plot(train.rpart)
```

To evaluate the accuracy of the classifier a prediction with testing data was performed.

```
> pred.rpart <- predict(train.rpart, newdata = testing)
> pred.rpart

 [1] 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1
Levels: 0 1
```

Classification-regression trees using this data set showed an accuracy of 85%, a sensitivity of 80% and a specificity of 90% to differentiate antimicrobial from non-antimicrobial peptides.

```
> confusionMatrix(data = pred.rpart, testing$group, positive = "1")
```

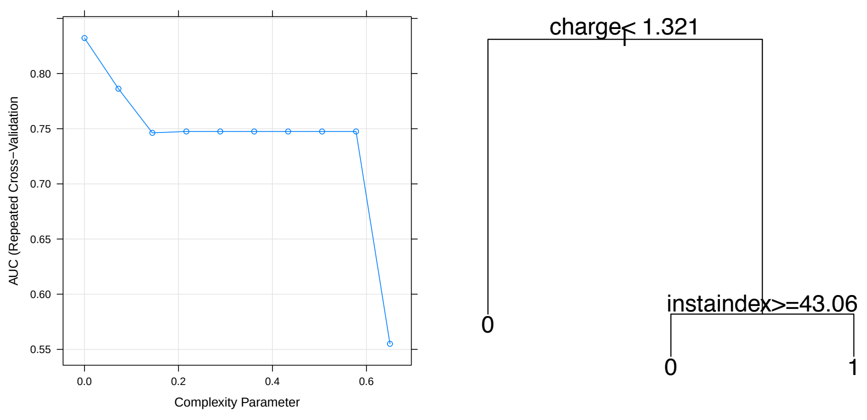


Figure 2: `plot(train.rpart)` shows the relationship between the complex parameter and the resampled estimate area under the ROC curve used to find the best model. `plot(tree)` shows the rules to classify the antimicrobial peptides of the Pepdata data set.

```
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 9 2
         1 1 8

               Accuracy : 0.85
                 95% CI : (0.6211, 0.9679)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.001288

                  Kappa : 0.7
 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.8000
            Specificity : 0.9000
         Pos Pred Value : 0.8889
         Neg Pred Value : 0.8182
             Prevalence : 0.5000
         Detection Rate : 0.4000
   Detection Prevalence : 0.4500
      Balanced Accuracy : 0.8500

       'Positive' Class : 1

> tree <- train.rpart$finalModel
> plot(tree, compress = TRUE, margin = c(0.1, 0.1, 0.1, 0.1), cex = 0.5)
> text(tree)
```

For this data set, the resulting classification-regression tree is based on characteristic properties of antimicrobial peptides such as charge and instability index (Figure 2).

The same process was conducted to evaluate the data potential using a linear discriminant analysis.

```
> train.lda <- train(group~., data = training,
+                    method = "lda",
+                    metric = "ROC",
+                    tuneLength = 10,
+                    trControl = fitControl)
> train.lda

Linear Discriminant Analysis

80 samples
```

```
21 predictors
 2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)

Summary of sample sizes: 72, 72, 72, 72, 72, 72, ...

Resampling results

  ROC       Sens  Spec  ROC SD     Sens SD    Spec SD
  0.821875  0.78  0.8   0.1534471  0.2169578  0.1978419

> pred.lda <- predict(train.lda, newdata = testing)
> pred.lda

 [1] 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1
Levels: 0 1
```

The linear discriminant analysis using this data set showed an accuracy of 95%, a sensitivity of 100% and a specificity of 90% to differentiate antimicrobial from non-antimicrobial peptides.

```
> confusionMatrix(data = pred.lda, testing$group,positive = "1")

Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0  9  0
         1  1 10

               Accuracy : 0.95
                 95% CI : (0.7513, 0.9987)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.003e-05

                  Kappa : 0.9
 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9000
         Pos Pred Value : 0.9091
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5500
      Balanced Accuracy : 0.9500

       'Positive' Class : 1
```

Summary

We introduced the **Peptides** package to calculate physicochemical properties and indices from amino acids sequences of peptides and proteins. We also showed how the package's functions can be used to produce useful data in classification-regression processes and handle the output files from the GROMACS molecular dynamics package. The authors are currently working on extending the package by the addition of new biological functions such as peptide flexibility (Huang and Nau, 2003), molecular lipophilicity potential (Gaillard et al., 1994), cell penetrating parameter (Holton et al., 2013) and free energy of binding (Hessa et al., 2007).

Acknowledgements

This project corresponds to a part of the first author's undergraduate thesis. We thank COLCIENCIAS from Colombia for the fellowship to PR and to the anonymous reviewers for their helpful comments

and suggestions.

Bibliography

F. C. Bernstein, T. F. Koetzle, G. J. Williams, E. F. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank: A computer-based archival file for macromolecular structures. *Archives of Biochemistry and Biophysics*, 185(2):584–591, 1978. [p9]

H. G. Boman. Antibacterial peptides: Basic facts and emerging concepts. *Journal of Internal Medicine*, 254(3):197–215, 2003. [p4, 7]

D. Eisenberg. Three-dimensional structure of membrane and surface proteins. *Annual Review of Biochemistry*, 53:595–623, 1984. [p7, 8]

D. Eisenberg, R. M. Weiss, and T. C. Terwilliger. The helical hydrophobic moment: A measure of the amphiphilicity of a helix. *Nature*, 299(5881):371–374, 1982. [p7, 8]

C. D. Fjell, J. Hiss, R. E. W. Hancock, and G. Schneider. Designing antimicrobial peptides: Form follows function. *Nature Reviews. Drug Discovery*, 11(1):37–51, 2012. [p4]

P. Gaillard, P.-A. Carrupt, B. Testa, and A. Boudon. Molecular lipophilicity potential, a tool in 3D QSAR: Method and applications. *Journal of Computer-Aided Molecular Design*, 8(2):83–96, 1994. [p12]

E. Gasteiger, C. Hoogland, A. Gattiker, M. R. Wilkins, R. D. Appel, A. Bairoch, et al. Protein identification and analysis tools on the ExPASy server. In *The Proteomics Protocols Handbook*, pages 571–607. Springer, 2005. [p4, 5, 7]

K. Guruprasad, B. V. Reddy, and M. W. Pandit. Correlation between stability of a protein and its dipeptide composition: A novel approach for predicting in vivo stability of a protein from its primary sequence. *Protein Engineering*, 4(2):155–61, 1990. [p6]

R. E. W. Hancock. Cationic peptides: Effectors in innate immunity and novel antimicrobials. *The Lancet. Infectious Diseases*, 1(3):153–164, 2001. [p4]

T. Hessa, N. M. Meindl-Beinker, A. Bernsel, H. Kim, Y. Sato, M. Lerch-Bader, I. Nilsson, S. H. White, and G. von Heijne. Molecular code for transmembrane-helix recognition by the Sec61 translocon. *Nature*, 450(7172):1026–1030, 2007. [p12]

T. A. Holton, G. Pollastri, D. C. Shields, and C. Mooney. CPPpred: Prediction of cell penetrating peptides. *Bioinformatics*, 29(23):3094–3096, 2013. [p12]

F. Huang and W. M. Nau. A conformational flexibility scale for amino acids in peptides. *Angewandte Chemie*, 115(20):2371–2374, 2003. [p12]

A. Ikai. Thermostability and aliphatic index of globular proteins. *Journal of Biochemistry*, 88(6): 1895–1898, 1980. [p6]

S. Kawashima, P. Pokarowski, M. Pokarowska, A. Kolinski, T. Katayama, and M. Kanehisa. AAindex: Amino acid index database, progress report 2008. *Nucleic Acids Research*, 36(suppl 1):D202–D205, 2008. [p7]

H. Kim, J. H. Jang, S. C. Kim, and J. H. Cho. De novo generation of short antimicrobial peptides with enhanced stability and cell specificity. *The Journal of Antimicrobial Chemotherapy*, 69(1):1–12, 2013. [p5]

M. Kuhn. *caret: Classification and Regression Training*, 2014. URL `http://CRAN.R-project.org/package=caret`. R package version 6.0-35. [p9]

K. Matsuzaki. Control of cell selectivity of antimicrobial peptides. *Biochimica et Biophysica Acta*, 1788 (8):1687–1692, 2009. [p4]

L. Monincová, J. Slaninová, V. Fučík, O. Hovorka, Z. Voburka, L. Bednárová, P. Maloň, J. Štokrová, and V. Čeřovskỳ. Lasiocepsin, a novel cyclic antimicrobial peptide from the venom of eusocial bee Lasioglossum laticeps (Hymenoptera: Halictidae). *Amino Acids*, 43(2):751–761, 2012. [p4]

D. S. Moore. Amino acid and peptide net charges: A simple calculational procedure. *Biochemical Education*, 13(1):10–11, 1985. [p6]

S. P. Piotto, L. Sessa, S. Concilio, and P. Iannelli. YADAMP: Yet another database of antimicrobial peptides. *International Journal of Antimicrobial Agents*, 39(4):346–351, 2012. [p4]

P. Rice, I. Longden, and A. Bleasby. EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics*, 16(6):276–277, 2000. [p4]

J. E. Stajich, D. Block, K. Boulez, S. E. Brenner, S. A. Chervitz, C. Dagdigian, G. Fuellen, J. G. Gilbert, I. Korf, H. Lapp, et al. The Bioperl toolkit: Perl modules for the life sciences. *Genome Research*, 12 (10):1611–1618, 2002. [p4]

S. Thomas, S. Karnik, R. S. Barai, V. K. Jayaraman, and S. Idicula-Thomas. CAMP: A useful resource for research on antimicrobial peptides. *Nucleic Acids Research*, 38:D774–780, 2010. [p4]

M. Torrent, D. Andreu, V. M. Nogués, and E. Boix. Connecting peptide physicochemical and antimicrobial properties by a rational prediction model. *PLoS ONE*, 6(2):e16968, 2011. [p6]

G. Wang, X. Li, and Z. Wang. APD2: The updated antimicrobial peptide database and its application in peptide design. *Nucleic Acids Research*, 37:D933–937, 2009. [p4, 9]

M. R. Yeaman and N. Y. Yount. Mechanisms of antimicrobial peptide action and resistance. *Pharmacological Reviews*, 55(1):27–55, 2003. [p4, 6]

Daniel Osorio
Escuela de Biología, Facultad de Ciencias
Universidad Industrial de Santander
Colombia
daniel.osorio@correo.uis.edu.co

Paola Rondón-Villarreal
Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones
Universidad Industrial de Santander
Colombia
paitorv@gmail.com

Rodrigo Torres
Escuela de Química, Facultad de Ciencias
Universidad Industrial de Santander
Colombia
rtorres@uis.edu.co

fanplot: An R Package for Visualising Sequential Distributions

by Guy J. Abel

Abstract Fan charts, first developed by the Bank of England in 1996, have become a standard method for visualising forecasts with uncertainty. Using shading fan charts focus the attention towards the whole distribution away from a single central measure. This article describes the basics of plotting fan charts using an R add-on package alongside some additional methods for displaying sequential distributions. Examples are based on distributions of both estimated parameters from a time series model and future values with uncertainty.

Introduction

Probabilities are notoriously difficult to communicate effectively to lay audiences (Spiegelhalter et al., 2011). Fan charts provide one such method to illustrate either forecasts or past results that are based on probabilistic distributions. Using shading fan charts focus the attention of the reader on the whole distribution away from a single central estimate. Visualising the distribution can aid in communicating the degree of underlying uncertainty in probabilistic forecasts to non-specialists, that might not have been apparent in basic plots and summary statistics.

Fan charts were first introduced by the Bank of England for their inflation forecasts in February 1996 (Britton et al., 1998). Since their initial development fan charts have become a standard method to display uncertainty of future economic indicators by many central banks (Julio, 2007). Their use has also spread to other fields such as climate science (McShane and Wyner, 2011) and demography (Gerland et al., 2014).

Fan charts can be created using various software. Within R, the **vars** package (Pfaff, 2008) has a `fanchart` function for forecasts of confidence regions. It is based solely on "varpred" class objects, i.e., on the predictions of Vector Autoregressive models fitted using other functions within the **vars** package. Similarly, the **forecast** package (Hyndman and Khandakar, 2008) produces fan charts for forecasts based on time series models from the "forecast" class. Julio (2009) provides VBA code in order to plot fan charts for quarterly GDP data in Excel. Alternatively one could use point and click methods in Excel to build customised fan charts based on stacked area charts.[1] Buchmann (2010) provides MATLAB code to create fan charts for user supplied forecast distributions with a limited amount of control for the plotted display.

In any of the fore-mentioned options users are restricted in either their ability to effectively adapt the properties of fan charts or create plots based on alternative models, values or simulated data. The aim of this article is to illustrate R code in the **fanplot** package to create fan charts of different styles and from a range of input data. These are demonstrated on data from sequential Monte Carlo Markov chain (MCMC) simulated distributions of parameters in a stochastic volatility model and expert based forecasts for the Consumer Price Index (CPI) of the Bank of England.

Fan charts for sequential simulated distributions

The **fanplot** package can used to display any form of sequential distributions along a plots x-axis. To illustrate, we use posterior density distributions of the estimated volatility of daily returns (y_t) from the Pound/Dollar exchange rate from 02/10/1981 to 28/6/1985. As Meyer and Yu (2002) show, posterior distributions for the volatility process can be estimated in WinBUGS by fitting the stochastic volatility model;

$$y_t|\theta_t = \exp\left(\frac{1}{2}\theta_t\right) u_t \qquad u_t \sim N(0,1) \qquad t = 1,\ldots,n. \tag{1}$$

The latent volatilities θ_t, which are unknown states in a state-space model terminology (Harvey, 1990), are assumed to follow a Markovian transition over time given by the state equations:

$$\theta_t|\theta_{t-1},\mu,\phi,\tau^2 = \mu + \phi \log \sigma^2_{t-1} + v_t \qquad v_t \sim N(0,\tau^2) \qquad t = 1,\ldots,n \tag{2}$$

with $\theta_0 \sim N(\mu,\tau^2)$.

[1]See `http://peltiertech.com/WordPress/excel-fan-chart-showing-uncertainty-in-projections` for an Excel tutorial.

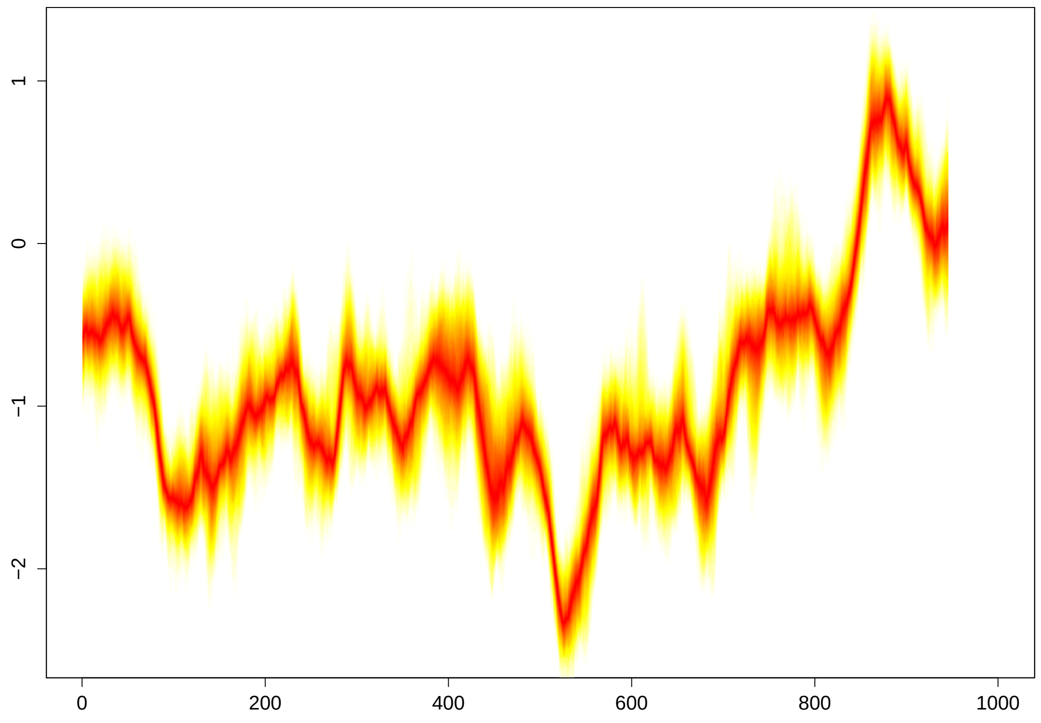


Figure 1: Stochastic Volatility Fan Chart for the Pound-Dollar Exchange Rate Data.

A sample of the posterior distributions of θ_t is contained in the `th.mcmc` object of the **fanplot** package. It consists of (1000) rows corresponding to MCMC simulations and (945) columns corresponding to time points t. Example code to replicate this object using **R2OpenBUGS** (Sturtz et al., 2005) is given in the help file for the `th.mcmc` object. It is based on the BUGS model of Meyer and Yu (2002) replicated in the `my1.R` file of the **fanplot** package. Time ordered simulated distributions, such as `th.mcmc`, can be easily extracted from the `sims.list` element of an **R2OpenBUGS** "bugs" object.

A fan chart of the evolution of the distribution of θ_t in Figure 1 can be plotted using either the `fan0` or `fan` function. The `fan0` function, which we will first discuss, provides the simplest representation;

```
library("fanplot")
fan0(data = th.mcmc)
```

The plotting function calculates the values of 100 equally spaced percentiles of each future distribution when the default `data.type = "simulations"` is set. This allows 50 fans to be plotted from the `heat.colours` colour palette, providing darker shadings for the more probable percentiles. The axis limits are determined from the `data` argument. By default, the y-axis limits to 85 percent of the range of the MCMC distributions to reduce white space in the plot.

Similar plots of sequential distributions from alternative Bayesian models can be easily plotted using the `fan0` or `fan` functions. The `data` argument accepts objects from a range of classes including "mcmc" which is typically used to handle BUGS or JAGS results via the `read.coda` or `read.jags` commands.

The data in `th.mcmc` are based on trading day observations only. Irregular time series can be handled by passing a **zoo** time series object (Zeileis and Grothendieck, 2005) to the data argument. The trading days are given in the spvdx object of the **tsbugs** package (Abel et al., 2013).

```
library("zoo")
library("tsbugs")

# create irregular multiple time series object
th.mcmc2 <- zoo(th.mcmc, order.by = svpdx$date)

# plot
fan0(data = th.mcmc2, type = "interval", ln = c(0.5, 0.8, 0.95),
     llab = TRUE, rcex = 0.6)
```

Basing the fan chart on a **zoo** time series allows the x-axis in Figure 2 to use the corresponding trading days rather than the observations index as in Figure 1. When argument `type = "interval"` is set,

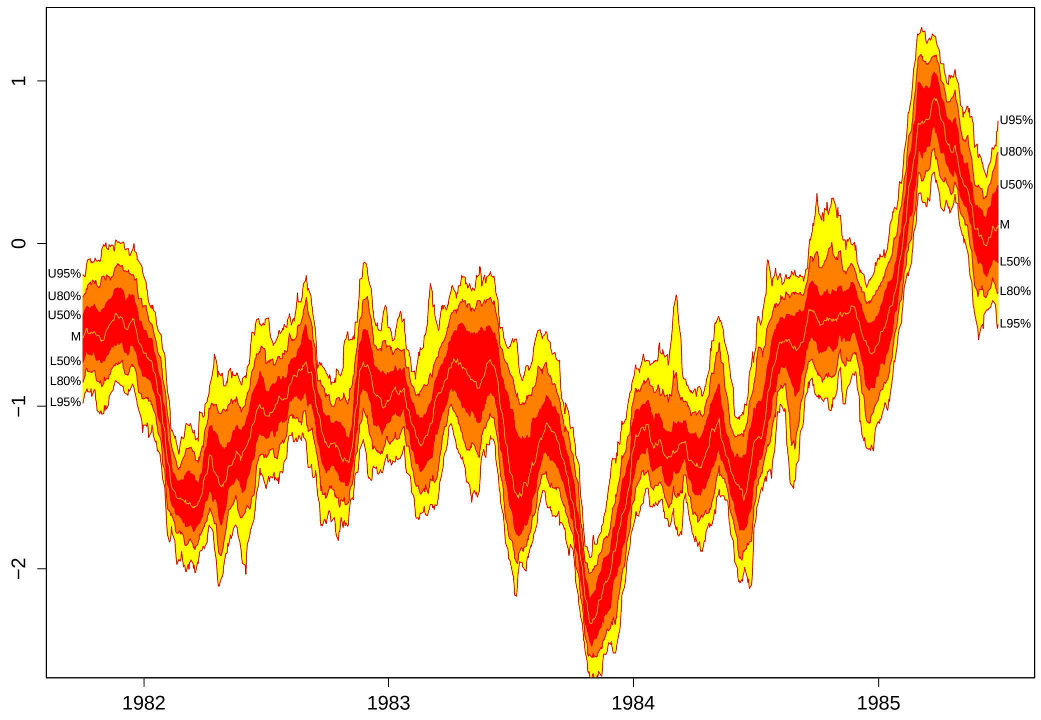


Figure 2: Stochastic Volatility Fan Chart for the Pound-Dollar Exchange Rate Data with Prediction Intervals.

the `probs` argument corresponds to prediction intervals. Consequently, the default interval fan chart comprises of three different shades, running from the darkest for the 50th prediction interval to the lightest for the 95th prediction interval. Contour lines are are controlled by the `ln` argument, which is set to `NULL` by default for `fan0`. In Figure 2, changes in the default of `ln` overlays lines on the fan chart for the upper and lower bounds of the 50th, 80th and 95th prediction intervals. A further line is plotted along the median of θ_t controlled by the `med.ln` argument and shown when `type = "interval"`. Labels on the right hand side are by default added to correspond to the upper and lower bounds of each plotted line. The text size of the labels are controlled by the `rcex` argument. This is set to 0.6 to incorporate the labels without extending the limits of the x-axis. The left labels are added by setting `llab = TRUE` and take the same text properties as the right labels. Users can customize many properties of the fan chart shading, labels contour lines, labels and axis through a range of arguments.

Spaghetti plots are a method of viewing data to visualize possible values through a systems. They are commonly used on geographical data, such as meteorological forecasts (Sanyal et al., 2010) to show possible or realised paths, or over time, such in longitudinal data analysis (Hedeker and Gibbons, 2006). Spaghetti plots can also be used represent uncertainty shown by a range of possible future trajectories or past estimates. For example, using the `th.mcmc2` object Figure 3 displays 20 random sets of θ_t simulations plotted by setting the argument `style = "spaghetti"`;

```
# transparent fan with visible lines
fan0(th.mcmc2, ln = c(5, 20, 50, 80, 95), alpha = 0, ln.col = "darkorange", llab = TRUE)

# spaghetti lines
fan(th.mcmc2, style = "spaghetti", n.spag = 20, alpha = 0.3)
```

The initial fan chart is completely transparent from setting the transparency argument `alpha = 0`. In order for the percentile lines to be visible a non-transparent colour is used for the `ln.col` argument. Lines are plotted according the user defined `ln` argument to provide underlying uncertainty measures for the posterior probability distribution. The spaghetti lines, which are semi-transparent, are based on a random selection of simulations. They are superimposed on a fan chart using the `fan` function, which operates in much the similar way as `fan0`. The most important difference between the two is in the default setting of the `add` argument, which controls whether to create a new plot window for a fan chart or add it to an existing device. For the `fan` function, `add` is set to `TRUE` and hence its is more appropriately used to add a fan chart to an existing plotting device. The `fan` function also adds lines and labels on select contours by default as illustrated in the next section.

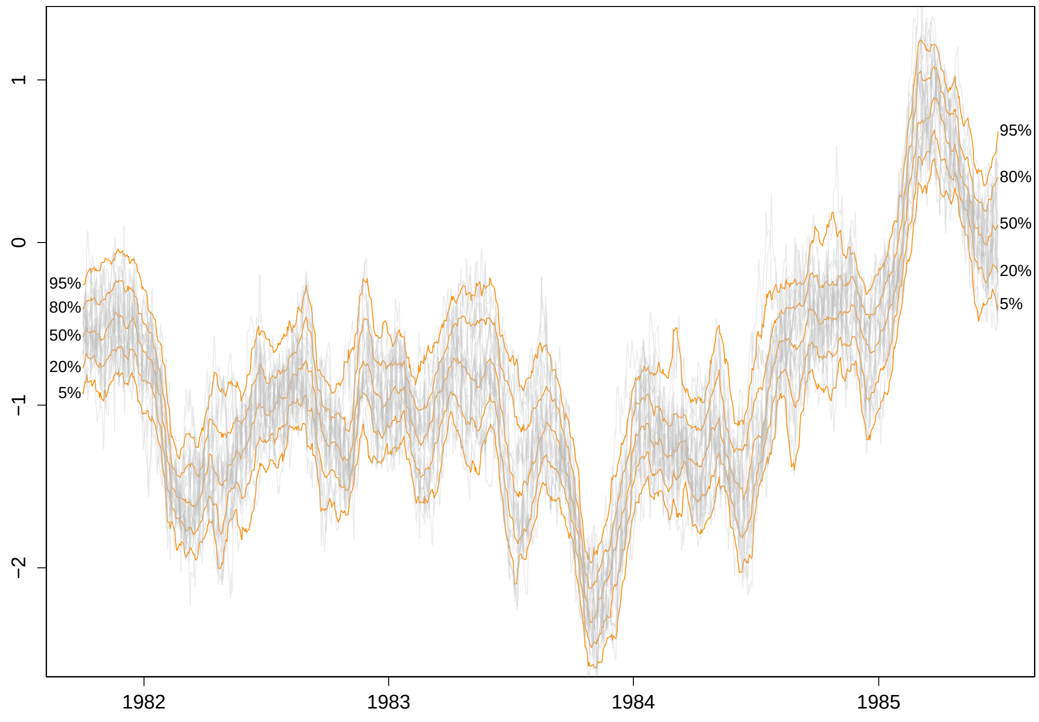


Figure 3: Stochastic Volatility Spaghetti Plot for the Pound-Dollar Exchange Rate Data.

Bank of England fan charts

The Monetary Policy Committee (MPC) of the Bank of England produces fan charts of forecasts for Consumer Price Index (CPI) of inflation and Gross Domestic Product in their quarterly Inflation Reports. Alongside the fan charts, the Bank of England provides data, in the form of central location, uncertainty and skewness parameters of a split-normal distribution that underlie their fan charts.[2] The probability density of the split-normal distribution is given by Julio (2007)[3] as,

$$f(x;\mu,\sigma_1,\sigma_2) = \begin{cases} \frac{\sqrt{2}}{\sqrt{\pi}(\sigma_1+\sigma_2)} e^{-\frac{1}{2\sigma_1^2}(x-\mu)^2} & \text{for } -\infty < x \leq \mu \\ \frac{\sqrt{2}}{\sqrt{\pi}(\sigma_1+\sigma_2)} e^{-\frac{1}{2\sigma_2^2}(x-\mu)^2} & \text{for } \mu < x < \infty \end{cases}, \qquad (3)$$

where μ represents the mode, and the two standard deviations σ_1 and σ_2 can be derived given the overall uncertainty parameter, σ and skewness parameters, γ, as;

$$\sigma^2 = \sigma_1^2(1+\gamma) = \sigma_2^2(1-\gamma). \qquad (4)$$

Functions for the probability density, cumulative distribution, quantiles and random generation for the split-normal distribution can be found in the **fanplot** package.

The boe data frame provides historical details on the forecasts of the MPC for CPI inflation between Q1 2004 to Q4 2013.

```
> head(boe)
  time0    time mode uncertainty skew
1  2004 2004.00 1.34      0.2249    0
2  2004 2004.25 1.60      0.3149    0
3  2004 2004.50 1.60      0.3824    0
4  2004 2004.75 1.71      0.4274    0
5  2004 2005.00 1.77      0.4499    0
6  2004 2005.25 1.68      0.4761    0
```

The first column `time0` refers to the base year of forecast, the second, `time` indexes future projections, whilst the remaining three columns provide values for the corresponding projected central location (μ), uncertainty (σ) and skew (γ) parameters:

[2]The Bank of England predominately refer to the equivalent, re-parametrised, two-piece normal distribution.
[3]Wallis (2014) notes the two-piece normal distribution was first introduced by Fechner (1897).

Bank of England style fan charts vary from quarter to quarter but follow a similar theme throughout, which can be replicated in R using the **fanplot** package. The input data given to a `fan` function to plot a fan chart differs from the simulations of the previous section. Rather than many simulations from distributions in each time period we can pass a "matrix" object of time ordered values from the split-normal quantile function.[4] As is the case for passing simulated values to the `fan` function, rows of the data object represent a set of user defined probabilities and columns represent a set of time points. For example, in the code below, a subset of the Bank of England future parameters of CPI published in Q1 2013 are first selected. Then a vector of probabilities related to the percentiles, that we ultimately would like to plot different shaded fans for, are created. Finally, in a `for` loop, the `qsplitnorm` function calculates the values for which the time-specific (`i`) split-normal distribution will be less than or equal to the probabilities of `p`.

```
# select relevant data
y0 <- 2013
boe0 <- subset(boe, time0 == y0)
k <- nrow(boe0)

# guess work to set percentiles the boe are plotting
p <- seq(0.05, 0.95, 0.05)
p <- c(0.01, p, 0.99)

# quantiles of split-normal distribution for each probability (row) at each future
# time point (column)
cpival <- matrix(NA, nrow = length(p), ncol = k)
for (i in 1:k)
   cpival[, i] <- qsplitnorm(p, mode = boe0$mode[i],
                             sd = boe0$uncertainty[i],
                             skew = boe0$skew[i])                                   #$
```

The new object `cpival` contains values evaluated from the `qsplitnorm` function in 6 rows, for our selected probabilities used in the calculation `p`, and 13 columns for successive time periods for which the MPC provide future parameters.

The `cpival` object can be used to add a fan chart to an active R graphic device. In the code below, the area of Figure 4 is set up when plotting the past CPI data, contained in the time series object `cpi`. The `xlim` arguments are set to ensure space on the right hand side of the plotting area for the fan. Following as closely as possible the Bank of England style for plotting fan charts for Q1 2013[5], the plotting area is set to near square, the background for future values is a gray colour, y-axis are plotted on the right hand side, a horizontal line are added for the CPI target and a vertical line for the two-year ahead point.

```
# past data
plot(cpi, type = "l", col = "tomato", lwd = 2,
     xlim = c(y0 - 5, y0 + 3), ylim = c(-2, 7),
     xaxt = "n", yaxt = "n", ylab = "")

# background shading during forecast period
rect(y0 - 0.25, par("usr")[3] - 1, y0 + 3, par("usr")[4] + 1,
     border = "gray90", col = "gray90")

# add fan
fan(data = cpival, data.type = "values", probs = p,
    start = y0, frequency = 4, anchor = cpi[time(cpi) == y0 - 0.25],
    fan.col = colorRampPalette(c("tomato", "gray90")), ln = NULL, rlab = NULL)

# boe aesthetics
axis(2, at = -2:7, las = 2, tcl = 0.5, labels = FALSE)
axis(4, at = -2:7, las = 2, tcl = 0.5)
axis(1, at = 2008:2016, tcl = 0.5)
axis(1, at = seq(2008, 2016, 0.25), labels = FALSE, tcl = 0.2)
abline(h = 2)                 # boe cpi target
abline(v = y0 + 1.75, lty = 2)  # 2 year line
```

[4]Note, values from any distributions quantile function can be used.

[5]The CPI fan chart for Q1 2013 can be viewed in the February 2013 Inflation Report (Bank of England, 2013), Chart 5.3.

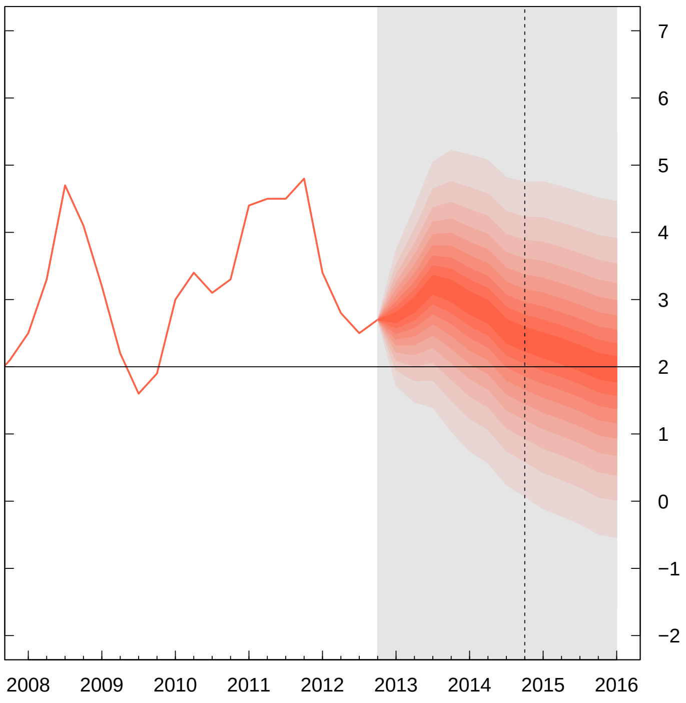


Figure 4: Fan chart, in the Bank of England style, for the MPC Q1 2013 forecast of the percentage increase in prices on year earlier.

The fan chart itself is outputted from the `fan` function, where arguments are set to ensure a close resemblance of the R plot to that produced by the Bank of England. The first three arguments in the `fan` function called in the above code, provide the `cpival` data to be plotted, indicate that the data are a set of calculated values (as opposed to simulations as in the previous examples) and provide the probabilities that correspond to each row of `cpival` object. The next two arguments define the start time and frequency of the data. These operate in a similar fashion to those used when defining time series in R with the `ts` function. The `anchor` argument is set to the value of CPI before the start of the fan chart. This allows a join between the value of the Q1 2013 observation and the fan chart. The `fan.col` argument is set to a colour palette for shades between `tomato` and `gray90`. The final two arguments are set to `NULL` to suppress the plotting of contour lines at the boundary of each shaded fan and their labels, as per the Bank of England style.

An alternative plot in Figure 5 is based on a regular time series object of simulated data and some other style settings in the `fan` function. These produce a fan chart with a greater array of coloured fans with labels and contour lines alongside selected percentiles of the future distribution. The input data is based upon 10,000 simulated values produced using the `rsplitnorm` function and the future split-normal distribution parameters from Q1 2013 in the truncated `boe0` data frame;

```
# simulate future values
cpisim <- matrix(NA, nrow = 10000, ncol = k)
for (i in 1:k)
  cpisim[, i] <- rsplitnorm(n = 10000, mode = boe0$mode[i],
                            sd = boe0$uncertainty[i],
                            skew = boe0$skew[i])
```

The fan chart based on the simulations in `cpisim` are then be added to the truncated CPI data plot;

```
# truncate cpi series and plot
cpi0 <- ts(cpi[time(cpi) < 2013], start = start(cpi), frequency = frequency(cpi))
plot(cpi0, type = "l", lwd = 2, las = 1, ylab = "",
     xlim = c(y0 - 5, y0 + 3.5), ylim = c(-2, 7))

# add fan
library("RColorBrewer")
fan(data = cpisim, type = "interval", probs = seq(0.01,0.99,0.01),
    start = y0, frequency = 4, ln = c(50,80,95), med.ln = FALSE,
    fan.col = colorRampPalette(colors = rev(brewer.pal(9, "Oranges"))))
```

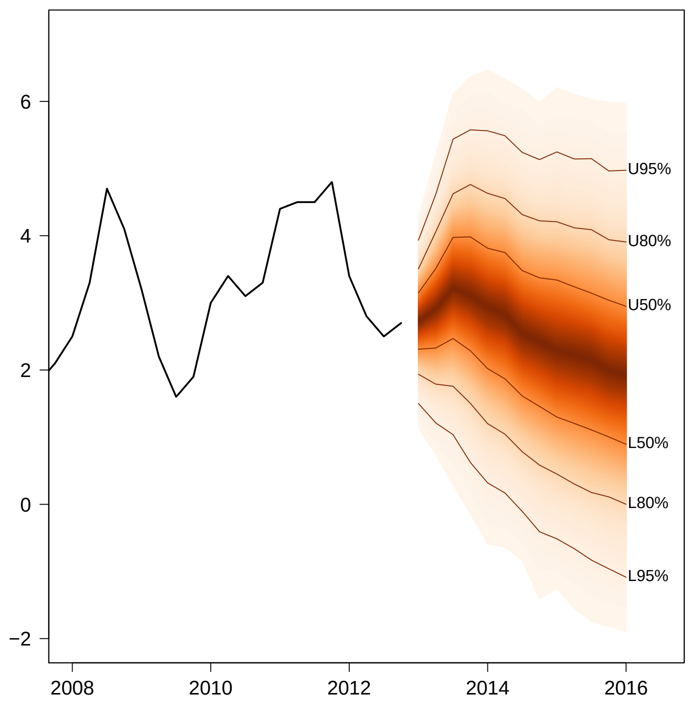


Figure 5: Alternative fan chart for the MPC Q1 2013 CPI forecast.

This code shows how users can control multiple visual elements of the fan chart, not previously illustrated. In Figure 5 the fan is based on 50 shadings for 100 equally spaced percentiles of the future distributions specified through the `probs` argument. The colour scheme is based on the `Oranges` palette from the **RColorBrewer** package (Neuwirth, 2014). Contour lines for the upper and lower intervals of the 50th, 80th and 95th prediction intervals are imposed using the `ln` argument. The median line, plotted by default when `type = "interval"` is set, is removed using the `med.ln` argument.

Box plots (Tukey, 1977) are commonly used as a simple descriptive statistics to visualise data through their quartiles. The `boxplot` function in R has many options including the display of multiple box plots based on sequential distributions. However, when data is based on a time series with multiple observations during a unit of time, such as quarterly data, fixing the location of the plot on the x-axis can be cumbersome. The `fan` function overcomes this problem when setting `style = "boxplot"`. In Figure 6 the simulated future CPI data `cpisim` are passed to the `data` argument:

```
# plot past data
plot(cpi0, type = "l", xlim = c(y0-5, y0+3), ylim = c(-2, 7), lwd = 2)

# box plots
fan(cpisim, style = "boxplot", start = y0, frequency = 4, outline = FALSE)
```

The `fan` function allows users to easily the locate sequential distributions of box plots on the x-axis using the `start` and `frequency` arguments. Additional arguments in the `fan` function are passed to `boxplot`. For example from the code above, outliers are suppressed by setting `outline = FALSE`.

Summary

The **fanplot** package allows users to easily visualise uncertainty based on either simulations from sequential distributions or values based on pre-calculated quartiles of distribution. Interactive visualisations of fan charts, as demonstrated in the `net_elicit.R` demo using the **shiny** package (RStudio and Inc., 2014), could potentially allow for an intuitive elicitation of experts forecasts[6]. Data of various classes can be incorporated including regular time series ("mts"), irregular time series ("zoo") and simulations from, say, MCMC via "matrix", "data.frame" or "mcmc" type objects. The fanplot package also has a range of options to adjust colour shadings of fan charts, their lines and labels, and specify whether to display prediction intervals or percentiles of distributions.[7]

[6]Run via `demo(net_elicit, package = "fanplot", ask = FALSE)`.

[7]An illustration of many of these options are provided in a demo file, run via `demo(sv_fan, package = "fanplot", ask = FALSE)`.

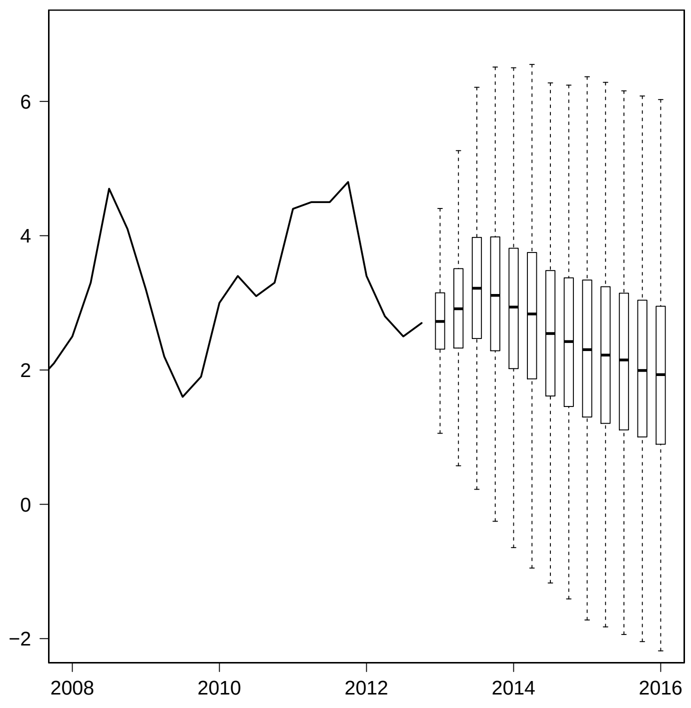


Figure 6: Box plots for the MPC Q1 2013 CPI forecast.

Bibliography

G. J. Abel, J. Bijak, J. J. Forster, J. Raymer, P. W. F. Smith, and J. S. Wong. Integrating uncertainty in time series population forecasts: An illustration using a simple projection model. *Demographic Research*, 29:1187–1226, Dec. 2013. [p16]

Bank of England. Inflation report February 2013. Technical report, Bank of England, London, United Kingdom, 2013. URL http://www.bankofengland.co.uk/publications/Documents/inflationreport/2013/ir13feb.pdf. [p19]

E. Britton, P. Fisher, and J. Whitley. The Inflation Report projections: understanding the fan chart. *Bank of England Quarterly Bulletin*, 38:30–37, 1998. [p15]

M. Buchmann. MATLAB file exchange: Fan chart, 2010. URL http://www.mathworks.de/matlabcentral/fileexchange/27702-fan-chart. [p15]

G. T. Fechner. Kollektivmasslehre. *Leipzig, Germany: Wilhelm Engelmann*, 1897. [p18]

P. Gerland, A. E. Raftery, H. Sevčíková, N. Li, D. Gu, T. Spoorenberg, L. Alkema, B. K. Fosdick, J. Chunn, N. Lalic, G. Bay, T. Buettner, G. K. Heilig, and J. Wilmoth. World population stabilization unlikely this century. *Science*, 387(6635):803–805, Sept. 2014. [p15]

A. C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, Cambridge, United Kingdom, 1990. [p15]

D. Hedeker and R. D. Gibbons. *Longitudinal Data Analysis*. John Wiley & Sons, Hoboken, NJ, USA, 2006. [p17]

R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 27:1–22, 2008. [p15]

J. M. Julio. The fan chart: The technical details of the new implementation. 2007. URL http://www.banrep.gov.co/docum/ftp/borra468.pdf. [p15, 18]

J. M. Julio. The HPD fan chart with data revision. 2009. URL http://mpra.ub.uni-muenchen.de/57714/. [p15]

B. B. McShane and A. J. Wyner. A statistical analysis of multiple temperature proxies: Are reconstructions of surface temperatures over the last 1000 years reliable? *The Annals of Applied Statistics*, 5(1): 5–44, Mar. 2011. [p15]

R. Meyer and J. Yu. BUGS for a Bayesian analysis of stochastic volatility models. *Econometrics Journal*, 3(2):198–215, Dec. 2002. [p15, 16]

E. Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL http://CRAN.R-project.org/package=RColorBrewer. [p21]

B. Pfaff. VAR, SVAR and SVEC models: Implementation within R package vars. *Journal of Statistical Software*, 27(4), 2008. [p15]

RStudio and Inc. *shiny: Web Application Framework for R*, 2014. URL http://CRAN.R-project.org/package=shiny. [p21]

J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. J. Moorhead. Noodles: a tool for visualization of numerical weather model ensemble uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1421–1430, 2010. [p17]

D. Spiegelhalter, M. Pearson, and I. Short. Visualizing uncertainty about the future. *Science*, 333(6048):1393–1400, Sept. 2011. [p15]

S. Sturtz, U. Ligges, and A. Gelman. R2WinBUGS: a package for running WinBUGS from R. *Journal of Statistical Software*, 12(3):1–16, 2005. [p16]

J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley Publishing Company, Reading, MA, USA, 1 edition, 1977. [p21]

K. F. Wallis. The two-piece normal, binormal, or double Gaussian distribution: Its origin and rediscoveries. *Statistical Science*, 29(1):106–112, Feb. 2014. [p18]

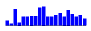
A. Zeileis and G. Grothendieck. zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 30(6):1–27, 2005. [p16]

Guy J. Abel
Wittgenstein Centre (IIASA, VID/ÖAW, WU)
Vienna Institute of Demography/Austrian Academy of Sciences
Wohllebengasse 12-14, 6th floor,
1040 Vienna, Austria
guy.abel@oeaw.ac.at

sparkTable: Generating Graphical Tables for Websites and Documents with R

by Alexander Kowarik, Bernhard Meindl and Matthias Templ

Abstract

Visual analysis of data is important to understand the main characteristics, main trends and relationships in data sets and it can be used to assess the data quality. Using the R package **sparkTable**, statistical tables holding quantitative information can be enhanced by including spark-type graphs such as sparklines and sparkbars .

These kind of graphics are well-known in literature and are considered as simple, intense and illustrative graphs that are small enough to fit in a single line. Thus, they can easily enrich tables and texts with additional information in a comprehensive visual way.

The R package **sparkTable** uses a clean S4 class design and provides methods to create different types of sparkgraphs that can be used in websites, presentations and documents. We also implemented an easy way for non-experts to create highly complex tables. In this case, graphical parameters can be interactively changed, variables can be sorted, graphs can be added and removed in an interactive manner. Thereby it is possible to produce custom-tailored graphical tables – standard tables that are enriched with graphs – that can be displayed in a browser and exported to various formats.

Introduction

Without doubt, visualization of data becomes more and more important for data analysis and for presenting aggregated information for end-users and policy needs. For surveys on the history and developments in visualization we refer to Friendly (2008) and Wickham (2013). The reason for this is that the human eye is a very powerful pattern recognition tool. Readers can easily understand simple visual presentations of data like charts published every day in newspapers, the internet, television and scientific articles. The visual presentation of data should illustrate trends and relationships quickly and easily and should display key figures about data. For readers it is often much easier to understand statistics when seeing a chart or a map rather than being confronted with a long list of numbers (see, e.g., The United Nations Economic Commission for Europe, 2009).

Cleveland and McGill (1987) state that information encoded in a graph depends on a number of aspects such as plotting symbols, length, slopes or colors. Changing any aspect of a graph possibly leads to a different perception and decoding of the information. Thus it is required to research aspects of graphical representation of data and finally come up with recommendations or best practices. This task has been been tackled by various authors (see, e.g., Cleveland and McGill, 1987; Tufte, 1986). Cleveland and McGill (1987) suggest, for example, an optimum mid-angle of 45 degrees when comparing the slopes of different lines.[1] Other suggestions on how to scale the data exist and are summarized in Heer and Agrawala (2006). Details on the choice of colors can be found in Zeileis et al. (2009).

Tables: Tables are a comprehensive way to summarize the most important facts about complex data. The United Nations Economic Commission for Europe (2009) claims that a small, well-formatted table can provide a great deal of information that readers can quickly absorb, and it, as well as Miller (2004), provide guidelines on how to design tables. However, the possibility of including graphics in tables (and text) is not covered.

Tufte's pioneer work: In our contribution, several examples show how to improve tables by including small graphics in the tables. It is pointed out that an improved table can contain various visual elements. This was originally proposed by Tufte (2001). He introduced the concept of sparklines and their application that is not limited to tables. These *intense, simple* and *word-sized graphics* sparklines can be embedded in continuous text and also match with other propositions he has made. For example, they can easily be used together with the small multiples approach (Tufte, 1990). According to Tufte it is always important to answer the question *compared to what?* when looking at graphs. Thus, the concept of small multiples is simply to put multiple graphs next to each other to allow direct visual comparisons between different (graphical) objects. He also states that small multiples are the best design solution to present a wide range of problems.

[1] For example, this is also considered in the sparklines embedded in Table 2.

Another proposition Tufte makes is to *show the data above all*. In order to do so he introduces the term of *data-ink ratio*. This ratio can be seen as the ink in a graph used to actually show data by the total ink used to print a graph. While the additional ink used for labels, axes, ticks and grids among others may help to improve graphs, it distracts attention from the most important part of the graph – the data values itself. Thus, in his view the data-ink ratio should be increased whenever possible and sparklines are a way to do so, since it is possible to show a high amount of data, trends and variations in small space. This concept has also been criticized. Inbar et al. (2007) evaluated people's acceptance of the minimalist approach (by increasing the data-ink ratio) to visualize information; a standard bar-graph and a minimalist version (both in Tufte, 2001) were compared. The majority of evaluators was more attracted by the standard graph. However, they didn't evaluate the case of placing sparklines in text or tables. Traditionally, the output in many areas of statistics is focused on tables (The United Nations Economic Commission for Europe, 2009), this is especially true for official statistics. Large tables with lots of number provide a lot of information, but make it hard to grasp it. Visualization can provide tools to make effects in the data visible at first sight and can provide the possibility to show more of the data than by just showing the numbers.

Software to produce sparklines: To make visualization methods applicable, software tools, which provide interactions with the data and provide reproducibility of any result, are needed.

To publish sparklines in newspapers has a long history. For example, the Huntsville Times reports the use of sparklines in their sports section on March 21, 2005. To produce this kind of sparklines and graphical tables, various software products are available. For example, SAS© and JMP© include such facilities.[2] Also Microsoft Excel allows to produce sparklines and there are implementations to produce simple sparklines for Photoshop, Prism, DeltaMaster and matplotlib. Most of these solutions are closed-source and commercial.

The sparkTable package

Our contribution goes a step beyond the before mentioned available tools. It is not just a re-implementation of sparklines in another software. Various improvements have been made and new features are available to the user.

The presented software is free and open-source, its features are designed in a well-specified, object-oriented manner with defined classes and methods. The presented software allows to save the sparklines in different output formats for easy inclusion in LaTeX or websites with many optional graphical features, i.e., the sparklines are highly customizable. The software features the simple generation of graphical tables with any statistical content that is automatically calculated from microdata. Finally, interactive features to create sparklines and sparktables are available and clickable in the browser.

In the following, a short introduction into the R package **sparkTable** (Kowarik et al., 2015) is given with the help of an example. The first version of the package was uploaded to the Comprehensive R Archive Network (CRAN) in May 2010 and has been improved various times since then.

The initial task was to find ways how common and frequent output such as tables and reports of statistical organizations could be improved using spark-type graphs and to find a technical solution to create these visualizations. Since R is gaining momentum in national statistical institutes all over the world, it was a natural choice to create an add-on package for R containing such functionality. There are a variety of reasons for this: one of them is the possibility to easily share and distribute the package and to instantly get feedback from users.

The goal was to create a free and open source software tool that allows the quick creation of spark-type graphics that could easily be included in various documents such as reports or web pages. Many examples for graphical tables have been produced and – based on the feedback from both colleagues and users – it seems as if graphical tables are an helpful tool to improve traditional, existing, tabular-based outputs of national statistical offices. Some of these graphical tables are presented below.

sparkTable can also be used by non experts in R. Interactive clickable features have been added to the package. Especially for graphical tables, some steps of abstraction are still required. However, it has been made easier to customize and change graphical tables by using interactive web applications. Instead of a formal description of classes and methods and internal implementation of the package, the functionality of the package is illustrated with examples.

[2]They offer also the link (batch call) to the **sparkTable** package (Hill, 2011).

Function	Description	Comment
`newSparkLine()` `newSparkBar()` `newSparkBox()` `newSparkHist()`	creation of sparklines, sparkbars, sparkboxplots and sparkhistograms that may serve as the base for creating graphical tables	creates objects of class 'sparkline', 'sparkbar', 'sparkbox' and 'sparkhist'
`plot()`	plots objects of class 'sparkline', 'sparkbar', 'sparkhist' or 'sparkbox'	
`export()`	saves objects of class 'sparkline', 'sparkbar', 'sparkhist', 'sparkbox' or 'sparkTable'	
`summaryST()`	summary for a data frame in a graphical table	creates an object of class 'sparkTable'. Customize the output with `setParameter()`
`newSparkTable()`	function to create an object of class 'sparkTable'	
`getParameter()`, `setParameter()`	basic functions to set parameters for objects of class 'sparkline', 'sparkbar', 'sparkbox', 'sparkTable' (or 'geoTable')	

Table 1: Most important functions of package **sparkTable**.

Command line usage of sparkTable by example

The first step is to install and load the package. We assume that R has already been started. Then the package can be simply installed from any CRAN server.

Help files can be accessed from R by typing `help(package = "sparkTable")` into the console. This results in a browse-able, linked help archive. Each method, function and data set that comes with the package is documented and also some example code is provided which allows users to become familiar with the package. So as to demonstrate the facilities of **sparkTable**, example data sets available in the package are used.

Table 1 shows the most important functions and classes of the **sparkTable** package. Basic functions are available to produce sparklines, e.g., `newSparkLine` to create an object of class 'sparkline' that can be used as input for the plotting method `plot()` which dispatches based on the class of the input method. Using method `export()` it is possible to save mini-plots as files (pdf, png or eps) on the hard-disk. In addition, functions are available to produce graphical tables and calculate summary statistics for tables, see Basic sparks and Graphical tables for practical examples and more explanations.

In the following we show how to create basic spark graphics in Basic sparks. Then we demonstrate how to produce graphical tables that include some sparks in Graphical tables. We finish by introducing the new interactive ways that can be used to change, modify and adjust objects generated with package **sparkTable** in Interactive features.

Basic sparks

The data set pop, which is included in the package, is used. This data set contains the Austrian population size for different age groups from 1981 to 2009. We start by loading the data set into the current R session.

```
> data(pop, package = "sparkTable")
```

This data set – available in long-format – contains three variables. The variable `time` describes the year, variable `variable` the different age groups and variable `value` the corresponding population numbers. Next, an object, which contains all the information required to plot a first sparkline, is generated. This can be achieved using function `newSparkLine()`. Calling the function and only setting argument `values` to use the population data of Austria stored in vector `pop_ges` as shown in the code listing below, we obtain a new object `sline` (respectively `slineIQR`, where also the Inter-Quartile-Range (IQR) is visualized) with default settings for a sparkline plot.

```
> pop_ges <- pop$value[pop$variable == "Insgesamt"]
> sline <- newSparkLine(values = pop_ges)
> slineIQR <- newSparkLine(values = pop_ges, showIQR = TRUE)
```

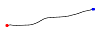
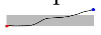
The resulting object `sline` is of class '`sparkline`' and has the default parameter settings. Later it is shown how to change specific settings of the graph using the generic function `setParameter()`. Note that it is also possible to directly change parameters using `newSparkLine()` directly. The help-files that can be accessed using `?newSparkLine` give an overview on the parameters that can be set or changed.

After creating a sparkline object it is possible to create a plot using function `plot()` or saving the graphic by applying method `export()`. Using these methods it is possible to plot and export not only objects of class '`sparkline`' but also objects of class '`sparkBar`' (generated with `newSparkBar()`), '`sparkBox`' (generated with `newSparkBox()`) and '`sparkHist`' (generated with `newSparkHist()`). The function `export()` requires a '`spark*`' object as input and allows to specify the desired output format. Currently, one of the following output formats can be selected by changing the function argument `outputType`:

- `"pdf"`: the spark-graphic is saved in pdf-format;
- `"eps"`: the graph is saved as an encapsulated postscript file;
- `"png"`: the graphic is written to disk as a portable network graph.

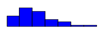
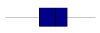
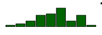
The function argument `outputType` is serialized so that it is possible to create the graphics in all possible formats in one step. By the help of argument `filename`, users can specify an output filename for the resulting graphic. The code listing below shows how to save the sparkline represented in obj `sline` as a pdf-graphic with filename `first-sl.pdf`.

```
> export(object = sline, outputType = "pdf",
+   filename = file.path("figures", "first-sl"))
> export(object = slineIQR, outputType = "pdf",
+   filename = file.path("figures", "first-sl-IQR"))
```

This graph can now easily be included in the current text showing the development  of the Austrian population from 1981 to 2009, alternatively with integrated IQR  .

Similarly, other types of graphics can be produced. The next code listing highlights how to create other supported kind of word-graphs. In addition, some of the default parameters are changed.

```
> v1 <- as.numeric(table(rpois(50, 2))); v2 <- rnorm(50)
> sbar <- newSparkBar(values = v1, barSpacingPerc = 5)
> sbox <- newSparkBox(values = v2, boxCol = c("black", "darkblue"))
> shist <- newSparkHist(values = v2, barCol = c("black", "darkgreen", "black"))
> export(object = sbar, outputType = "pdf",
+   filename = file.path("figures", "first-bar"))
> export(object = sbox, outputType = "pdf",
+   filename = file.path("figures", "first-box"))
> export(object = shist, outputType = "pdf",
+   filename = file.path("figures", "first-hist"))
```

Since they were exported as pdf-files, the produced  sparkbar, the  sparkbox and the  sparkhist can be easily integrated into a LaTeX-document as shown here.

Changing parameters of spark objects can also be achieved by using function `setParameter()`. In the help file (accessible via `?setParameter`) all possible values that can be changed are explained and listed. Some of the options that might be changed are now listed below:

- `width`: available vertical space of the graph;
- `height`: available horizontal space of the graph;
- `values`: data points that should be plotted;
- `lineWidth`: the thickness of the line in a sparkline-plot;

- `pointWidth`: the radius of special points (minimum, maximum, last observation) of a sparkline;
- `showIQR`: whether the IQR is plotted in a sparkline-graph;
- `outCol`: color of outlying observations in a sparkbox-graph;
- `barSpacingPerc`: percent of the total horizontal space used between the bars in a sparkbar-graph.

We now show how to change some graphical parameters of the spark graphics we have just produced.

```
> sline2 <- setParameter(sline, type = "lineWidth", value = 2)
> sline2 <- setParameter(sline2, type = "pointWidth", value = 5)
> export(object = sline2, outputType = "pdf",
+   filename = file.path("figures", "second-sl"))
```

For example the thickness of the line was changed in the sparkline object `sline2` from the default value of 1 to 2. Additionally, we changed the size of the points for special values in the sparkline to 5. If this line is plotted, the sparkline showing Austrian population numbers looks different.

It is worth noting that the current settings can be queried using function `getParameter()` in an analogue way as setting parameters by just leaving out the argument `values`. Detailed information about the kind of information that can be extracted using this function is available from the help page that is accessible via `?getParameter`.

Further, it is worth noting that these mini-graphs can also be directly included in fully automated reports. In the following example we show how to include a sparkline into a HTML-document that is created with **knitr** (Xie, 2014b) and the *R markdown* language.

~~~
```{r, echo=TRUE}
require(sparkTable)
sl <- newSparkLine(values = rnorm(25), lineWidth = .18, pointWidth = .4,
  width = .4, height = .08)
export(sl, outputType = "png", filename = "sparkLine")
```
This is a sparkline included in the ![firstSparkLine](sparkLine.png)
text...
~~~

Listing 1: Content of minimal markdown file that contains a sparkline.

A minimal example is shown in Listing 1. In the first part, a sparkline graph with random values is generated and saved as an image in png-format. We made sure that the height of the image equals approximately the height of a single line. Using markdown code, this image is finally included in the text. The integration in LaTeX , using `Sweave`, **knitr** or **brew** (Horner, 2011), is straightforward, i.e., the code environment in Listing 1 needs to be changed to `Sweave`, **knitr** or **brew** style.

Graphical tables

In Basic sparks, the basic usage of the package **sparkTable** to produce sparklines was demonstrated. Now we show how to enrich tables by including sparkgraphs. In addition, it is shown how sparkgraphs can be used to visualize regional indicators in so-called checkerplots (Templ et al., 2013). We point out that existing tools – specifically **xtable** (Dahl, 2014) – are used to generate the final output.

In order to plot a graphical table, the first step is to create a suitable input object - in this case an object of class '`sparkTable`'. Objects of this kind hold the data itself in a suitable format. Further, such objects contain a list with information about the type of graphs or functions on the data values that should be shown in the resulting table. Moreover, a vector of variable names must be listed to specify for which variables in the data input graphs should be produced. These steps can be achieved using function `newSparkTable()`. To illustrate this procedure, we give an example using results from the 2013/2014 soccer season in Austria.

```
> # load data (in long format)
> data(AT_Soccer, package = "sparkTable")
> # first three observations to see the structure of the data
> head(AT_Soccer, 3)

   team time points wl goaldiff shotgoal getgoal
1 Rapid    1      1  0        0        2       2
2 Rapid    2      3  1        4        4       0
3 Rapid    3      3  1        2        4       2
```

```
> # prepare content
> content <- list(
+     function(x) { sum(x) },
+     function(x) { round(sum(x), 2) },
+     function(x) { round(sum(x), 2) },
+     newSparkLine(lineWidth = 2, pointWidth = 6), newSparkBar()
+ )
> names(content) <- c("Points", "ShotGoal", "GetGoal", "GoalDiff", "WinLose")
>
> # set variables
> vars <- c("points", "shotgoal", "getgoal", "goaldiff", "wl")
>
> # create the sparkTable object
> stab <- newSparkTable(dataObj = AT_Soccer, tableContent = content, varType = vars)
```

The required code to create a 'sparkTable' object as shown above can be splitted into four sections. In the first line the input data set (in suitable long format for function newSparkTable()) is loaded.

The next step is to prepare a list specifying the content of each column of the required table. In this case the table is specified by five columns. In the first three columns very simple user-defined functions are used to aggregate three different variables. Note that the provided function can be arbitrarily complex.

For the last two columns of the table, sparklines and sparkbars are created. In this case we show that it is also possible to change parameters directly when creating such objects. We used this possibility to change the thickness of the line and the size of points for sparklines as it was already described in Basic sparks. It should also be noted that the names of this input list correspond to column names in the resulting final table, therefore we set the names of this list to the desired column headers. For example, the first column in the final table will have column name *Points*.

After the object stab has been established, the graphical table can be created. The required code is shown below:

```
> export(stab, outputType = "tex", filename = file.path("figures", "first-stab"),
+   graphNames = file.path("figures", "first-stab"))
```

The function export() is applied on the object stab, which has just been created in the previous example. The format of the output can be specified. By setting argument outputType to 'tex', the functions write LaTeX-code in the file specified with argument filename. Additionally, some instructions on how to include the table into a document are printed in the R session. It is worth to note that the code used to generate the table is also returned to R, which makes it easy to include it into reproducible reports. The final result of this example is shown in Table 2. The embedded sparklines in the second last column of Table 2 present an optimum mid-angle of 45 degrees of the slopes of each time series.

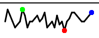
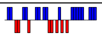
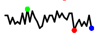
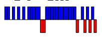
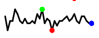
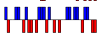
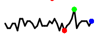
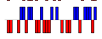
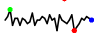
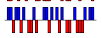
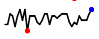
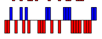
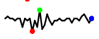
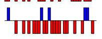
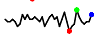
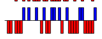
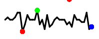
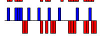
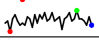
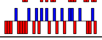
	Points	ShotGoal	GetGoal	GoalDiff	WinLose
Rapid	62	63	40		
Salzburg	80	110	35		
Austria	53	58	44		
Sturm	48	55	56		
Groedig	54	68	71		
Neustadt	39	43	84		
FC Wacker	29	42	70		
Wolfsberg	41	51	63		
Ried	43	55	66		
Admira	42	51	67		

Table 2: A graphical table featuring data values and sparkgraphs applied to Austrian soccer data. For each game in the entire season, wins are maked blue, defeats in read and no bars are plotted for draws. In addition we present differences in goals throughout the season an well as overall points and goals for each team.

In case a user wants to include a table in LaTeX, a short tex-macro (*graph*), which also has to be included in the LaTeX source, is printed by default in the R console. This information can be switched off by setting function argument infonote to FALSE.

Geographical tables

The concept to create geographical tables is based on the concept of the so called checkerplot (Templ et al., 2013), which is also implemented in the package **sparkTable** in the function `checkerplot()`. We now show how to generate a geographical table for the EU using the function `newGeoTable()` and `plotGeoTable()`. The final result is shown in Table 3. Let us first load some data and print the first three rows of the data:

```
> data(popEU, package = "sparkTable")
> data(debtEU, package = "sparkTable")
> data(coordsEU, package = "sparkTable")
> popEU <- popEU[popEU$country %in% coordsEU$country, ]
> debtEU <- debtEU[debtEU$country %in% coordsEU$country, ]
> EU <- cbind(popEU, debtEU[, -1])
> head(EU, 3)
```

```
  country    B1999    B2000    B2001    B2002    B2003    B2004    B2005
1      BE 10213752 10239085 10263414 10309725 10355844 10396421 10445852
2      BG  8230371  8190876  8149468  7891095  7845841  7801273  7761049
3      CZ 10289621 10278098 10266546 10206436 10203269 10211455 10220577
     B2006    B2007    B2008    B2009    B2010 X1999 X2000 X2001 X2002
1 10511382 10584534 10666866 10753080 10839905 113.7 107.9 106.6 103.5
2  7718750  7679290  7640238  7606551  7563710  77.6  72.5  66.0  52.4
3 10251079 10287189 10381130 10467542 10506813  16.4  18.5  24.9  28.2
  X2003 X2004 X2005 X2006 X2007 X2008 X2009 X2010
1  98.5  94.2  92.1  88.1  84.2  89.6  96.2  96.8
2  44.4  37.0  27.5  21.6  17.2  13.7  14.6  16.2
3  29.8  30.1  29.7  29.4  29.0  30.0  35.3  38.5
```

Before we generate sparklines, the data has to be transformed to long format. `reshapeExt()` can be used to bring the data to the form so that `newSparkTable()` or `newGeoTable()` can be applied. In `reshapeExt()`, parameter `idvar` defines variables in long format that identify multiple records from the same group/individual and `v.names` corresponds to names of variables in the long format that correspond to multiple variables in the wide format, see `?reshapeExt` for details. We show the first two list elements of the reordered data:

```
> EUlong <- reshapeExt(EU, idvar = "country", v.names = c("pop", "debt"),
+   varying = list(2:13, 14:25), geographicVar = "country", timeValues = 1999:2010)
> head(EUlong, 2)
```

```
$BE
      country time      pop  debt
BE.1       BE 1999 10213752 113.7
BE.2       BE 2000 10239085 107.9
BE.3       BE 2001 10263414 106.6
BE.4       BE 2002 10309725 103.5
BE.5       BE 2003 10355844  98.5
BE.6       BE 2004 10396421  94.2
BE.7       BE 2005 10445852  92.1
BE.8       BE 2006 10511382  88.1
BE.9       BE 2007 10584534  84.2
BE.10      BE 2008 10666866  89.6
BE.11      BE 2009 10753080  96.2
BE.12      BE 2010 10839905  96.8

$BG
      country time     pop debt
BG.1       BG 1999 8230371 77.6
BG.2       BG 2000 8190876 72.5
BG.3       BG 2001 8149468 66.0
BG.4       BG 2002 7891095 52.4
BG.5       BG 2003 7845841 44.4
BG.6       BG 2004 7801273 37.0
BG.7       BG 2005 7761049 27.5
BG.8       BG 2006 7718750 21.6
BG.9       BG 2007 7679290 17.2
```

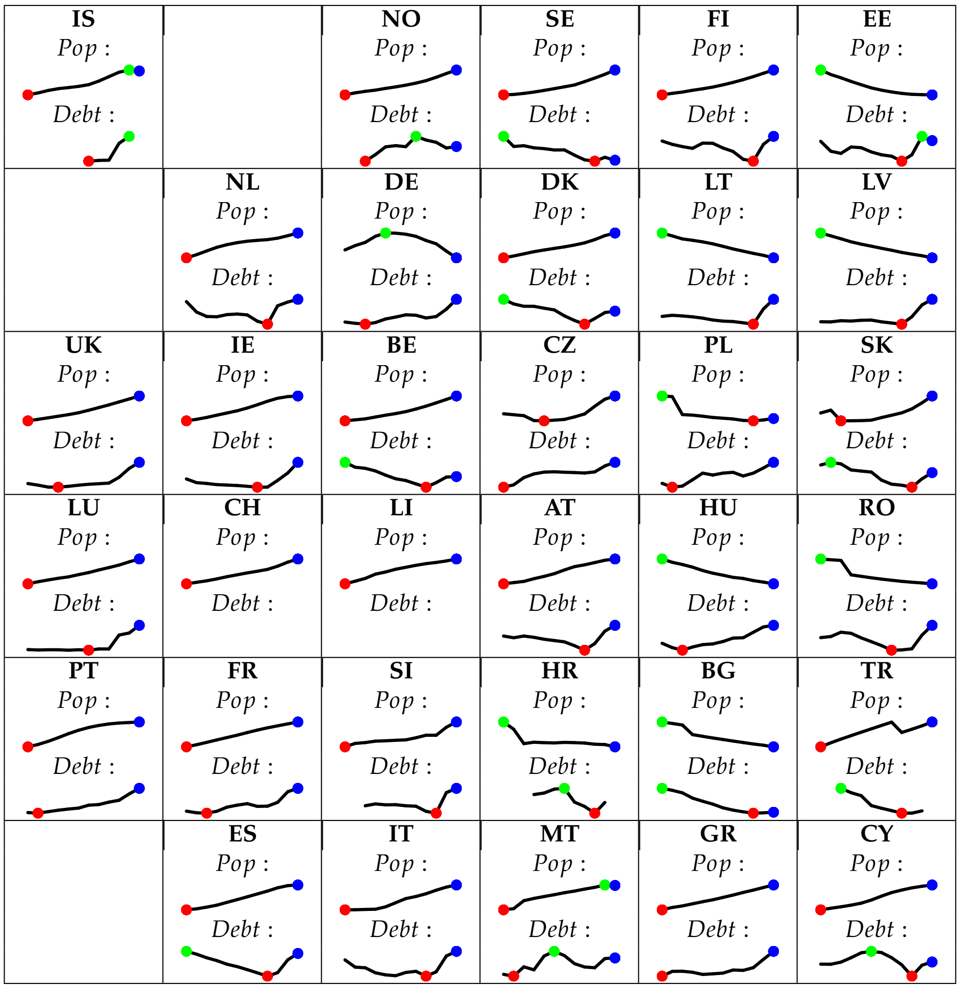


Table 3: A geographical table featuring sparkgraphs for all EU countries.

```
BG.10       BG 2008 7640238 13.7
BG.11       BG 2009 7606551 14.6
BG.12       BG 2010 7563710 16.2
```

The data is now ready and we can produce the geographical table:

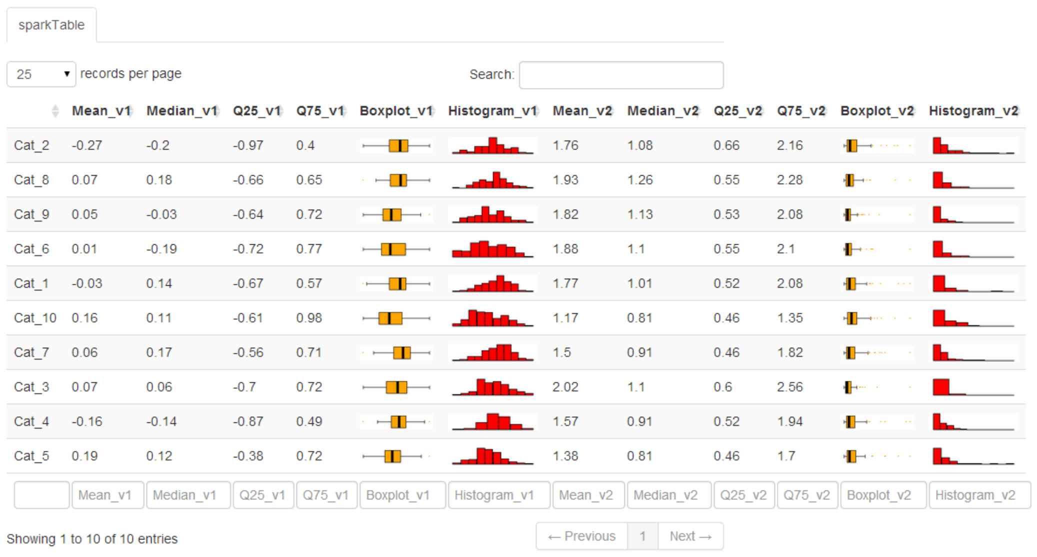
```
> l <- newSparkLine(lineWidth = 3, pointWidth = 10)
> content <- list(function(x) { "Population:" }, l, function(x) {"Debt:" }, l)
> varType <- c(rep("pop", 2), rep("debt", 2))
> xGeoEU <- newGeoTable(EUlong, content, varType, geographicVar = "country",
+   geographicInfo = coordsEU)
> export(xGeoEU, outputType = "tex", graphNames = file.path("figures", "out1"),
+   filename =  file.path("figures", "testEUT"), transpose = TRUE)
```

For the geographical representation in a grid, geographical coordinates have to be provided. Internally, an allocation problem is solved by linear programming to find the nearest free grid cell from the provided coordinates under several constraints. For this task, we refer to Templ et al. (2013), resulting in, for example, Scandinavian countries placed in the north, south Mediterranean countries in the south, etc. Again, the sparklines in each grid can be modified by function `setParameter()`.

Interactive features

Until recently **sparkTable** only provided rigid views of graphical tables in documents and on homepages. However, when including a graphical table on a homepage some interactivity is beneficial. Two functions using the functionality from **shiny** (Chang et al., 2015) allow for interactive graphical tables.

View your graphical table in a shiny app: showSparkTable() is a function to generate a shiny app to view your graphical table directly in a browser with some basic interactive functions like sorting and filtering. At the moment, methods for objects of class 'sparkTable' and 'data.frame' already are implemented. Figure 1 shows the output of the function applied to a 'sparkTable' object.



	Mean_v1	Median_v1	Q25_v1	Q75_v1	Boxplot_v1	Histogram_v1	Mean_v2	Median_v2	Q25_v2	Q75_v2	Boxplot_v2	Histogram_v2
Cat_2	-0.27	-0.2	-0.97	0.4			1.76	1.08	0.66	2.16		
Cat_8	0.07	0.18	-0.66	0.65			1.93	1.26	0.55	2.28		
Cat_9	0.05	-0.03	-0.64	0.72			1.82	1.13	0.53	2.08		
Cat_6	0.01	-0.19	-0.72	0.77			1.88	1.1	0.55	2.1		
Cat_1	-0.03	0.14	-0.67	0.57			1.77	1.01	0.52	2.08		
Cat_10	0.16	0.11	-0.61	0.98			1.17	0.81	0.46	1.35		
Cat_7	0.06	0.17	-0.56	0.71			1.5	0.91	0.46	1.82		
Cat_3	0.07	0.06	-0.7	0.72			2.02	1.1	0.6	2.56		
Cat_4	-0.16	-0.14	-0.87	0.49			1.57	0.91	0.52	1.94		
Cat_5	0.19	0.12	-0.38	0.72			1.38	0.81	0.46	1.7		
	Mean_v1	Median_v1	Q25_v1	Q75_v1	Boxplot_v1	Histogram_v1	Mean_v2	Median_v2	Q25_v2	Q75_v2	Boxplot_v2	Histogram_v2

Showing 1 to 10 of 10 entries

Figure 1: Shiny output of a 'sparkTable' object.

In the following, the key features of customizeSparkTable() are discussed.

Tab 1: Import sparkTable objects

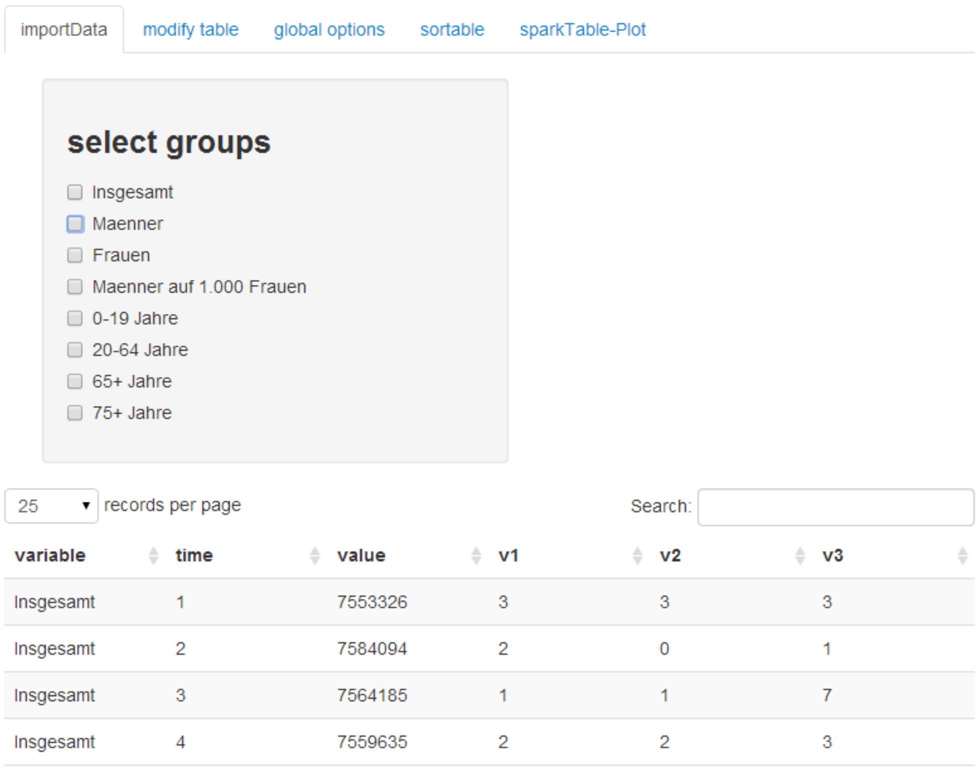
The provided app is organized in tabs. In each of the five tabs, specific changes can be made. Figure 2 shows the first tab – *importData* – that is available to the user when the function is called for the first time. We point out that this function has to be started with a 'sparkTable' object as input. In the upper part of this page, all possible groups that are available in the input data object are listed. In the lower part of the page an interactive table is shown. By using the mouse to toggle check-boxes it is possible to choose the groups that should be included in the final output.

Tab 2: Preparation of the table

The second tab – *modify table* – that is shown in Figure 3 is also partitioned into two parts. In the upper part, all the columns that are currently present in the output table are listed. It is then possible to select columns that should be deleted. After selecting the corresponding columns by toggling the check boxes and clicking on the *remove selected columns*-button, these columns are dropped from the table. However, in this tab it is also possible to add an additional column to the table. In this case the user needs to provide a column name and can select both the type of the column and the variable, which should be used, by selecting values from a drop-down menu. All supported types of graphs (*sparkline*, *sparkbar*, *sparkhist* and *sparkbox*) are available as well as the special type *function*, which can be used to calculate some statistics of the selected variable. An example is the determination of the maximal data value.

Tab 3: Global options

In the *global options*-tab (see Figure 4) it is possible to change the specifics of the current output. For each column, a block of options is listed. The user has the option to change the column name by modifying the current heading in a text box. It is also possible to change the type of a column by selecting a new type from a drop-down box or to adjust graphical parameters (such as point sizes or line width or colors) for some plots. For columns that are of type *function*, the function definition can be altered. For the adjustment of graphical parameters, suitable inputs such as sliders or drop-down boxes are used. After the desired modification has been applied, the user is required to press the modify-button for the currently changed column. The page will refresh and additional changes can be made. We note, however, that for the time being it is only possible to change one variable after another.



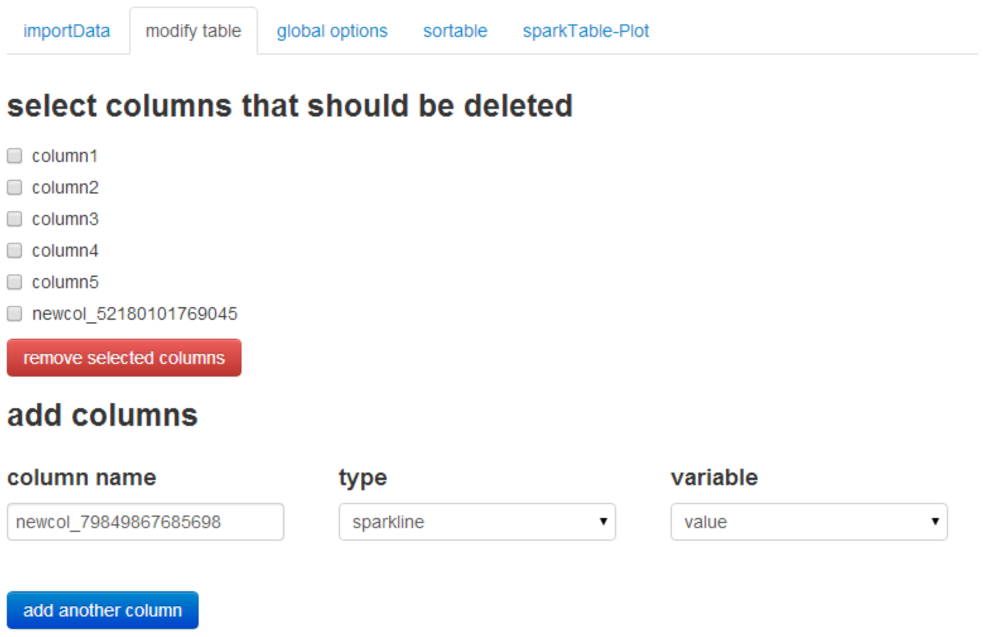
select groups

- [ ] Insgesamt
- [ ] Maenner
- [ ] Frauen
- [ ] Maenner auf 1.000 Frauen
- [ ] 0-19 Jahre
- [ ] 20-64 Jahre
- [ ] 65+ Jahre
- [ ] 75+ Jahre

25 records per page   Search:

variable	time	value	v1	v2	v3
Insgesamt	1	7553326	3	3	3
Insgesamt	2	7584094	2	0	1
Insgesamt	3	7564185	1	1	7
Insgesamt	4	7559635	2	2	3

Figure 2: Data options applicable for a 'sparkTable' object.



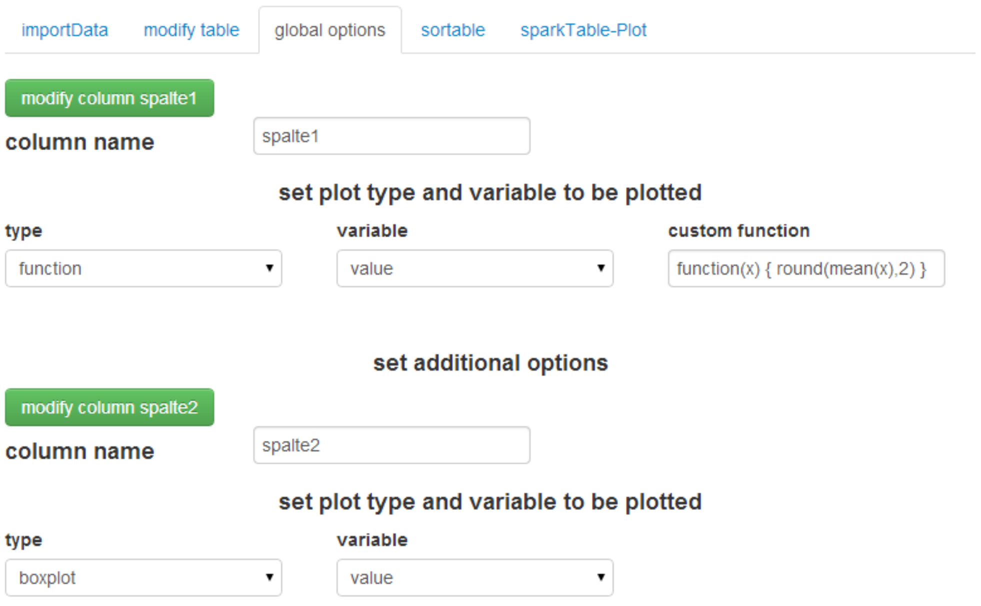
select columns that should be deleted

- [ ] column1
- [ ] column2
- [ ] column3
- [ ] column4
- [ ] column5
- [ ] newcol_52180101769045

add columns

column name	type	variable
newcol_79849867685698	sparkline	value

Figure 3: Possibilities for modifications of the loaded 'sparkTable' object.



column name: spalte1

set plot type and variable to be plotted

type: function
variable: value
custom function: function(x) { round(mean(x),2) }

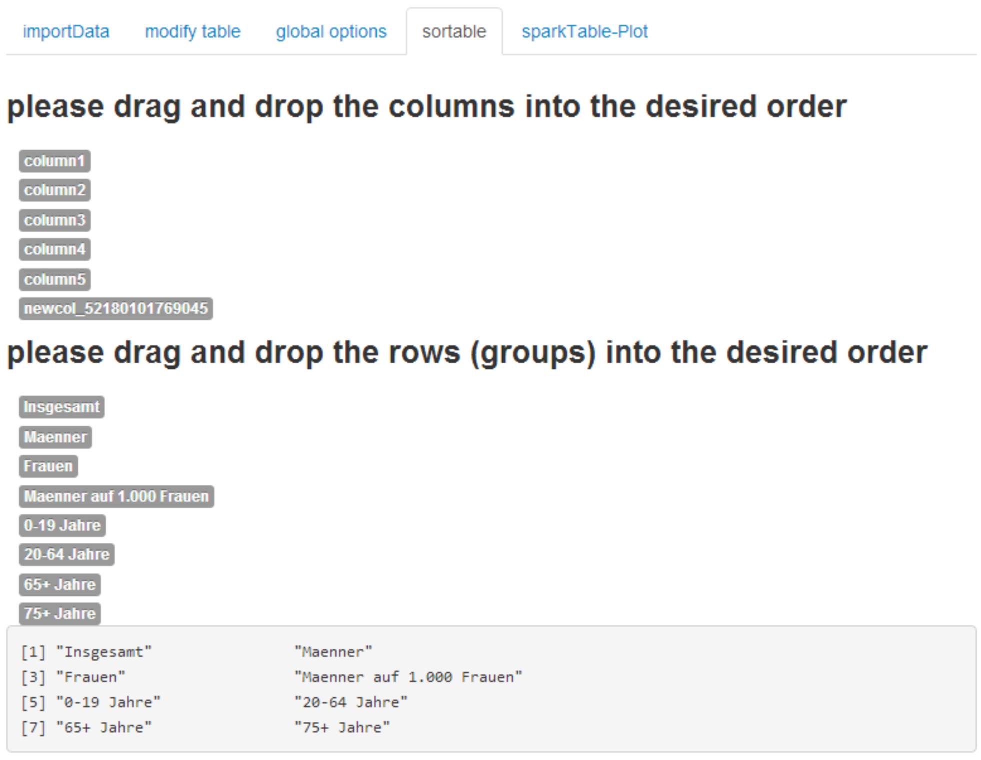
set additional options

column name: spalte2

set plot type and variable to be plotted

type: boxplot
variable: value

Figure 4: Global options.



please drag and drop the columns into the desired order

column1
column2
column3
column4
column5
newcol_52180101769045

please drag and drop the rows (groups) into the desired order

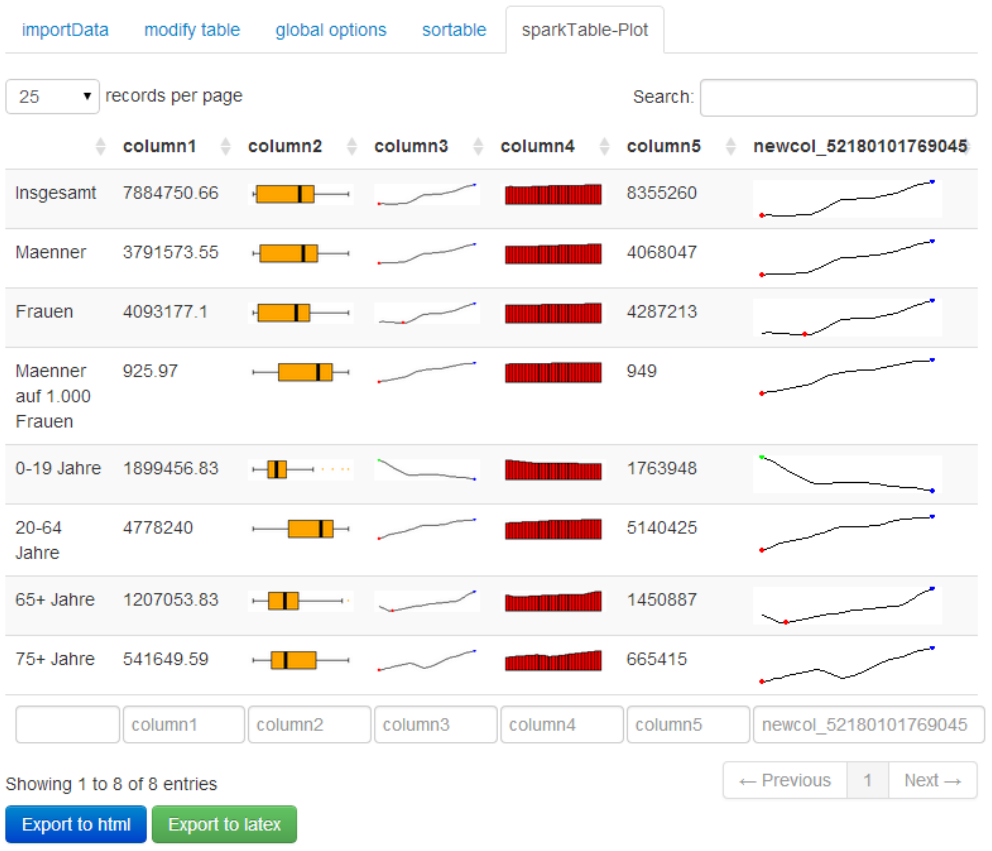
Insgesamt
Maenner
Frauen
Maenner auf 1.000 Frauen
0-19 Jahre
20-64 Jahre
65+ Jahre
75+ Jahre

```
[1] "Insgesamt"               "Maenner"
[3] "Frauen"                  "Maenner auf 1.000 Frauen"
[5] "0-19 Jahre"              "20-64 Jahre"
[7] "65+ Jahre"               "75+ Jahre"
```

Figure 5: Sort variables and groups in the sparkTable.

Tab 4: Ordering of columns

In the fourth tab (*sortable*), which is shown in Figure 5, it is possible to interactively change the ordering of columns (in the first part of the page) and rows (in the second part of the page) by using the mouse to drag boxes around until the desired ordering has been reached. The current result is automatically applied to the 'sparkTable' object, which is always visible in the last tab.



	column1	column2	column3	column4	column5	newcol_52180101769045
Insgesamt	7884750.66				8355260	
Maenner	3791573.55				4068047	
Frauen	4093177.1				4287213	
Maenner auf 1.000 Frauen	925.97				949	
0-19 Jahre	1899456.83				1763948	
20-64 Jahre	4778240				5140425	
65+ Jahre	1207053.83				1450887	
75+ Jahre	541649.59				665415	
	column1	column2	column3	column4	column5	newcol_52180101769045

Showing 1 to 8 of 8 entries

Figure 6: Preview of the output sparkTable.

Tab 5: The graphical table

In the tab *sparkTable-Plot* (see Figure 6), the current graphical table with all the current options is displayed. At the bottom of the page the user is offered two buttons. Using *Export to html*, the graphical table can be exported as an HTML-file, whereas pressing the *Export to latex*-button exports the final output into tex-format. Furthermore, the required R code to create the current graphical table is also printed to the current R session. Thus, users may adjust, copy or reuse a suitable plot-layout that was created with the interactive procedure.

Conclusions

Eduard R. Tufte's quote "Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space" holds especially true for graphical tables embedding sparkgraphs, since a lot of information can be put into a table with small but comprehensive (spark) graphs.

We gave a few examples of sparklines and graphical tables, for example, on sparkbars and sparklines in graphical tables to visualize trends and performances in sport results (compare Figure 2). Further obvious applications for sparklines are, e.g., economic and financial data. With the help of such sparklines, it is possible to present graphs, which allow to track and compare trends on changes in time series. Statistical tables can also be enhanced with such types of graphs.

Histograms and box-plots presented in graphical tables can be used to show the distribution of data together with some basic statistics in any publication that present empirical results or analyses in general. Sparkbars can thereby also be used to visualize frequency counts on categories of a variable. For specific summaries of data using sparkboxplots and sparkbars, see for example Hron et al. (2013). Also tabulated information for policy needs can be embedded in graphical tables.

The package **sparkTable** provides a flexible and interactive way to produce graphical tables for printed or web publication. Graphical tables have the key advantage to provide more information and insight in the same amount of space. A strict class oriented implementation using S4 classes organizes the internal production of graphical tables and sparklines in a pre-defined manner. The users are provided with highly customizable functions to create sparklines and tables in a flexible manner. The sparklines can be modified easily and the tables can also contain arbitrarily comples statistics based on the underlying data.

Additionally, interactivity is added with the help of the package **shiny**. Especially these interactive features allow non-experts in R to create graphical tables enriched by all kind of sparklines. Tables can then be exported to HTML or LaTeX and additionally the code required for various plots is shown both when the interactive shiny-app is used and when graphical tables are exported with `export()`.

Bibliography

W. Cleveland and R. McGill. Graphical perception: The visual decoding of quantitative information on graphical displays of data. *Journal of the Royal Statistical Society. Series A (General)*, 150(3):192–229, 1987. [p24]

D. B. Dahl. *xtable: Export Tables to LaTeX or HTML*, 2014. URL `http://CRAN.R-project.org/package=xtable`. R package version 1.7-3. [p28]

M. Friendly. A brief history of data visualization. In C. Chen, W. Härdle, and A. Unwin, editors, *Handbook of Data Visualization*, pages 15–56. Springer-Verlag, Heidelberg, 2008. [p24]

J. Heer and M. Agrawala. Multi-scale banking to 45 degrees. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):701–708, Sept. 2006. [p24]

E. Hill. JMP© 9 add-ins: Taking visualization of SAS© data to new heights. In *SAS Global Forum 2011*, 2011. [p25]

J. Horner. *brew: Templating Framework for Report Generation*, 2011. URL `http://CRAN.R-project.org/package=brew`. R package version 1.0-6. [p28]

K. Hron, M. Templ, and P. Filzmoser. Estimation of a proportion in survey sampling using the logratio approach. *Metrika*, 76(6):799–818, 2013. [p35]

O. Inbar, N. Tractinsky, and J. Meyer. Minimalism in information visualization: Attitudes towards maximizing the data-ink ratio. In *Proceedings of the 14th European Conference on Cognitive Ergonomics: Invent! Explore!*, ECCE '07, pages 185–188, New York, NY, USA, 2007. [p25]

A. Kowarik, B. Meindl, and M. Templ. *sparkTable: Sparklines and Graphical Tables for TEX and HTML*, 2015. URL `http://CRAN.R-project.org/package=sparkTable`. R package version 1.0.0. [p25]

J. Miller. *The Chicago Guide to Writing about Numbers*. Chicago Guides to Writing, Editing, and. University of Chicago Press, 2004. [p24]

RStudio and Inc. *shiny: Web Application Framework for R*, 2014. URL `http://CRAN.R-project.org/package=shiny`. R package version 0.9.1. [p31, 140]

M. Templ, B. Hulliger, A. Kowarik, and K. Fürst. Combining geographical information and traditional plots: The checkerplot. *International Journal of Geographical Information Science*, 27(4):685–698, 2013. [p28, 30, 31]

The United Nations Economic Commission for Europe. *Making Data Meaningful. Part 2: A Guide to Presenting Statistics*, 2009. ECE/CES/STAT/NONE/2009/3. [p24, 25]

E. Tufte. *Envisioning Information*. Graphics Press, 1990. [p24]

E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, USA, 1986. [p24]

E. R. Tufte. *The Visual Display of Quantitative Information*. Bertrams, 2nd edition, 2001. [p24, 25]

H. Wickham. Statistical graphics. *Encyclopedia of Environmetrics*, 2013. Pre-print available at `http://vita.had.co.nz/papers/stat-graphics.html`. [p24]

Y. Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2014. URL `http://CRAN.R-project.org/package=knitr`. R package version 1.6. [p28, 99]

A. Zeileis, K. Hornik, and P. Murrell. Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, 53(9):3259–3270, 2009. [p24]

Alexander Kowarik
Statistics Austria
Guglgasse 13; 1110 Vienna
Austria
alexander.kowarik@statistik.gv.at

Bernhard Meindl
Statistics Austria
Guglgasse 13; 1110 Vienna
Austria
bernhard.meindl@statistik.gv.at

Matthias Templ
CSTAT - Computational Statistics
Institute of Statistics & Mathematical Methods in Economics
Vienna University of Technology & Statistics Austria
Wiedner Hauptstr. 7; 1040 Vienna
Austria
matthias.templ@tuwien.ac.at

rdrobust: An R Package for Robust Nonparametric Inference in Regression-Discontinuity Designs

by Sebastian Calonico, Matias D. Cattaneo and Rocío Titiunik

Abstract This article describes the R package **rdrobust**, which provides data-driven graphical and inference procedures for RD designs. The package includes three main functions: `rdrobust`, `rdbwselect` and `rdplot`. The first function (`rdrobust`) implements conventional local-polynomial RD treatment effect point estimators and confidence intervals, as well as robust bias-corrected confidence intervals, for average treatment effects at the cutoff. This function covers sharp RD, sharp kink RD, fuzzy RD and fuzzy kink RD designs, among other possibilities. The second function (`rdbwselect`) implements several bandwidth selectors proposed in the RD literature. The third function (`rdplot`) provides data-driven optimal choices of evenly-spaced and quantile-spaced partition sizes, which are used to implement several data-driven RD plots.

Introduction

The regression-discontinuity (RD) design is a widely employed quasi-experimental research design in social, behavioral and related sciences; for reviews see Imbens and Lemieux (2008) and Lee and Lemieux (2010). In this design, units are assigned to treatment based on whether their value of an observed covariate is above or below a known cutoff, and the probability of receiving treatment conditional on this covariate jumps discontinuously at the cutoff. This jump induces "variation" in treatment assignment that may be regarded, under appropriate assumptions, as being unrelated to potential confounders. Thus, inference in RD designs is typically conducted using only observations near the cutoff or threshold, where the discontinuous change in the probability of treatment assignment occurs. Due to its local nature, RD average treatment effects estimators are usually constructed using local-polynomial nonparametric regression, and statistical inference is based on large-sample approximations.

This article gives an introduction to the R package **rdrobust** (Calonico et al., 2015b), which offers an array of data-driven local-polynomial and partitioning-based inference procedures for RD designs. We introduce three main functions implementing several data-driven nonparametric point and confidence intervals estimators, bandwidth selectors, and plotting procedures useful for RD empirical applications:

- `rdrobust()`. This function implements the bias-corrected robust (to "large" bandwidth choices) inference procedure proposed by Calonico, Cattaneo, and Titiunik (2014a, CCT hereafter), as well as many other RD inference procedures employing local-polynomial regression. The function `rdrobust` offers bias-corrected robust confidence intervals for average treatment effects at the cutoff for sharp RD, sharp kink RD, fuzzy RD and fuzzy kink RD designs.
- `rdbwselect()`. This function implements several data-driven bandwidth selectors for RD designs based on the recent work of Imbens and Kalyanaraman (2012, IK hereafter) and CCT. Although this command may be used as a stand-alone bandwidth selector in RD applications, its main purpose is to provide fully data-driven bandwidth choices to be used by our main function `rdrobust()`.
- `rdplot()`. This function implements several data-driven optimal choices of evenly-spaced and quantile-spaced bins, which are useful to produce RD plots that either approximate the regression function by local sample averages or represent the overall variability of the data in a disciplined way. These optimal choices are based on an integrated mean squared error expansion of appropriately constructed partitioning estimators, as discussed in Calonico, Cattaneo, and Titiunik (in press); see also Cattaneo and Farrell (2013) for related results. These binned sample means and partition size chosen are used to construct the popular RD plots commonly found in RD applications in a fully automatic way.

We first provide a brief review of all the methods implemented in **rdrobust**, and then discuss an empirical illustration using some of the features of our functions `rdrobust()`, `rdbwselect()` and `rdplot()`. A full description of the capabilities of the package **rdrobust** is available in its manual and help files. A companion Stata (StataCorp., 2013) package is described in Calonico, Cattaneo, and Titiunik (2014b).

Review of methods

We now present the basic RD framework, describe the population parameters of interest, introduce the local-polynomial based estimators, review different inference procedures, and briefly summarize the popular RD plots. Regularity conditions and other technical aspects underlying the estimands and estimators may be found in the references given throughout. Recent results and further details on RD designs are discussed in IK and CCT (see also the supplemental appendix), and references therein.

Setup and notation

We adopt the potential outcomes framework commonly employed in the treatment effects literature (e.g., Heckman and Vytlacil 2007 and Imbens and Wooldridge 2009). Let $\{(Y_i(0), Y_i(1), T_i(0), T_i(1), X_i)' : i = 1, 2, \ldots, n\}$ be a random sample from $(Y(0), Y(1), T(0), T(1), X)'$, where $Y(1)$ and $Y(0)$ denote the potential outcomes with and without treatment, respectively, $T(0)$ and $T(1)$ denote potential treatment status, and the scalar regressor $X_i \in \mathbb{R}$ is the so-called "running variable" or "score" determining treatment assignment based on whether it exceeds a known cutoff. In particular, unit i is assigned treatment if $X_i > \bar{x}$ and not assigned treatment if $X_i < \bar{x}$, for some known fixed value $\bar{x} \in \mathbb{R}$. $f(x)$ denotes the continuous (Lebesgue) density of X_i.

This setup allows for imperfect compliance, which in the RD literature is known as the *fuzzy RD design*. The case of perfect treatment compliance is usually called the *sharp RD design*. In either case, the observed outcome and treatment status are, respectively,

$$Y_i = \begin{cases} Y_i(0) & \text{if } X_i < \bar{x} \\ Y_i(1) & \text{if } X_i \geq \bar{x} \end{cases} \qquad \text{and} \qquad T_i = \begin{cases} T_i(0) & \text{if } X_i < \bar{x} \\ T_i(1) & \text{if } X_i \geq \bar{x} \end{cases} .$$

The observed data is $\{(Y_i, T_i, X_i)' : i = 1, 2, \ldots, n\}$, a random sample from a large population. In sharp RD designs, $T_i = \mathbb{1}(X_i \geq \bar{x})$ with $\mathbb{1}(\cdot)$ denoting the indicator function, which leads to $\mathbb{P}[T_i = 0|X_i < \bar{x}] = 1 = \mathbb{P}[T_i = 1|X_i \geq \bar{x}]$. More generally, in fuzzy RD designs, treatment assignment and treatment status may differ (imperfect compliance). Therefore, for each unit i, the scalar random variable $Y_i \in \mathbb{R}$ denotes the outcome of interest, and $T_i \in \{0, 1\}$ denotes actual treatment take-up ($T_i = 1$ treatment taken, $T_i = 0$ treatment not taken).

We introduce some additional notation. For $\nu \in \mathbb{Z}_+ = \{0, 1, 2, \ldots\}$, define

$$\mu^{(\nu)}_{Y+}(\bar{x}) = \lim_{x \to \bar{x}^+} \frac{\partial^\nu}{\partial x^\nu} \mu_{Y+}(x), \qquad \mu^{(\nu)}_{Y-}(\bar{x}) = \lim_{x \to \bar{x}^-} \frac{\partial^\nu}{\partial x^\nu} \mu_{Y-}(x),$$

with $\mu_{Y+}(x) = \mathbb{E}[Y(1)|X = x]$ and $\mu_{Y-}(x) = \mathbb{E}[Y(0)|X = x]$, and

$$\mu^{(\nu)}_{T+}(\bar{x}) = \lim_{x \to \bar{x}^+} \frac{\partial^\nu}{\partial x^\nu} \mu_{T+}(x), \qquad \mu^{(\nu)}_{T-}(\bar{x}) = \lim_{x \to \bar{x}^-} \frac{\partial^\nu}{\partial x^\nu} \mu_{T-}(x),$$

with $\mu_{T+}(x) = \mathbb{E}[T(1)|X = x]$ and $\mu_{T-}(x) = \mathbb{E}[T(0)|X = x]$. Whenever there is no ambiguity, we drop the subindex denoting the dependent variable or the point of evaluation in the conditional expectations and other functions.

Population parameters of interest

We focus on average treatment effects at the cutoff in the sharp RD, fuzzy RD, sharp kink RD and fuzzy kink RD designs, although the results cover other possibilities. For further details on the interpretation of these estimands and regularity conditions see, among others, Hahn et al. (2001), Porter (2003), Lee (2008), IK, CCT, Card et al. (2014), and references therein.

- **Sharp RD designs**. Two popular parameters of interest are the sharp RD average treatment effect at the threshold, denoted by τ_0, and the sharp kink RD average treatment effect at the threshold, denoted by τ_1/κ, with κ a known scale constant, and using the generic notation:

  $$\tau_\nu := \tau_{Y,\nu}(\bar{x}) = \frac{\partial^\nu}{\partial x^\nu} \mathbb{E}[Y_i(1) - Y_i(0)|X_i = x]\bigg|_{x=\bar{x}} = \mu^{(\nu)}_{Y+} - \mu^{(\nu)}_{Y-}, \qquad \nu \in \mathbb{Z}_+,$$

  where the definition drops the subindex denoting the dependent random variable and evaluation point for notational simplicity. The last equality is a nonparametric identification result that holds under mild continuity conditions: the parameter τ_ν can be written as a function of observed data because $\mu^{(\nu)}_{Y+} = \mu^{(\nu)}_{Y+}(\bar{x})$ and $\mu^{(\nu)}_{Y-} = \mu^{(\nu)}_{Y-}(\bar{x})$ are estimable from the observed data; $\tau_\nu = \tau_{Y,\nu}$ is a difference of two (one-sided) nonparametric regression functions at $\bar{x}$.

- **Fuzzy RD designs**. The parameters of interest take the form

$$\varsigma_\nu := \varsigma_\nu(\bar{x}) = \frac{\frac{\partial^\nu}{\partial x^\nu}\mathbb{E}[Y_i(1) - Y_i(0)|X_i = x]\Big|_{x=\bar{x}}}{\frac{\partial^\nu}{\partial x^\nu}\mathbb{E}[T_i(1) - T_i(0)|X_i = x]\Big|_{x=\bar{x}}} = \frac{\tau_{Y,\nu}}{\tau_{T,\nu}} = \frac{\mu^{(\nu)}_{Y+} - \mu^{(\nu)}_{Y-}}{\mu^{(\nu)}_{T+} - \mu^{(\nu)}_{T-}}, \qquad \nu \in \mathbb{Z}_+,$$

where the last two equalities represent a nonparametric identification result that holds under mild continuity conditions. Note that $\mu^{(\nu)}_{Y+} = \mu^{(\nu)}_{Y+}(\bar{x})$, $\mu^{(\nu)}_{Y-} = \mu^{(\nu)}_{Y-}(\bar{x})$, $\mu^{(\nu)}_{T+} = \mu^{(\nu)}_{T+}(\bar{x})$ and $\mu^{(\nu)}_{T-} = \mu^{(\nu)}_{T-}(\bar{x})$ are all estimable from the observed data, since $\tau_{Y,\nu}$ and $\tau_{T,\nu}$ are each a difference of two (one-sided) regression functions at $\bar{x}$, and ς_ν is just their ratio. In the RD literature, the two main parameters of interest are ς_0, the fuzzy RD average treatment effect at the cutoff, and ς_1, the fuzzy kink RD average treatment effect at the cutoff.

Local polynomial estimators

Statistical inference in the RD design reduces to nonparametric regression inference at the induced boundary point $\bar{x}$, employing observations at either side of the threshold separately. Local-polynomial estimators have become the preferred choice of nonparametric estimator in the RD literature because of their excellent boundary properties (Fan and Gijbels, 1996). The package **rdrobust** implements local polynomial estimators of various orders for the two variables Y_i and T_i depending on the RD design considered, and also includes different bandwidths selectors and alternative confidence intervals estimators. All these features are briefly reviewed in the following subsections, but first we introduce the local polynomial RD estimators of order p in general. To reduce repetition, we describe the estimators using a generic outcome variable Z which either takes the value Y or T, depending on the outcome variable under consideration. For $Z \in \{Y, T\}$ and $\nu, p \in \mathbb{Z}_+$ with $\nu \le p$,

$$\hat{\tau}_{Z,\nu,p}(x; h_n) = \hat{\mu}^{(\nu)}_{Z+,p}(x; h_n) - \hat{\mu}^{(\nu)}_{Z-,p}(x; h_n),$$

$$\hat{\mu}^{(\nu)}_{Z+,p}(x; h_n) = \mathbf{e}'_\nu \hat{\boldsymbol{\beta}}_{Z+,p}(x; h_n) \qquad \text{and} \qquad \hat{\mu}^{(\nu)}_{Z-,p}(x; h_n) = \mathbf{e}'_\nu \hat{\boldsymbol{\beta}}_{Z-,p}(x; h_n),$$

$$\hat{\boldsymbol{\beta}}_{Z+,p}(x; h_n) = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \sum_{i=1}^{n} \mathbb{1}(X_i \ge x)(Z_i - \mathbf{r}_p(X_i - x)'\boldsymbol{\beta})^2 K_{h_n}(X_i - x),$$

$$\hat{\boldsymbol{\beta}}_{Z-,p}(x; h_n) = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \sum_{i=1}^{n} \mathbb{1}(X_i < x)(Z_i - \mathbf{r}_p(X_i - x)'\boldsymbol{\beta})^2 K_{h_n}(X_i - x),$$

where here $\mathbf{r}_p(x) = (1, x, \ldots, x^p)'$, $\mathbf{e}_\nu$ is the conformable $(\nu + 1)$-th unit vector (e.g., $\mathbf{e}_1 = (0, 1, 0)'$ if $p = 2$), $K_h(u) = K(u/h)/h$ with $K(\cdot)$ a kernel function and h_n is a positive bandwidth sequence.

Using the generic notation above, the sharp RD estimators are:

$$\hat{\tau}_{\nu,p}(h_n) := \hat{\tau}_{Y,\nu,p}(\bar{x}; h_n), \qquad \nu \le p.$$

Similarly, for the fuzzy RD designs we have the RD estimators:

$$\hat{\varsigma}_{\nu,p}(h_n) := \frac{\hat{\tau}_{Y,\nu,p}(h_n)}{\hat{\tau}_{T,\nu,p}(h_n)}, \qquad \hat{\tau}_{Y,\nu,p}(h_n) := \hat{\tau}_{Y,\nu,p}(\bar{x}; h_n), \qquad \hat{\tau}_{T,\nu,p}(h_n) := \hat{\tau}_{T,\nu,p}(\bar{x}; h_n), \qquad \nu \le p.$$

Assuming the bandwidth $h_n \to 0$ and other regularity conditions hold, consistency of these estimators follows easily from well-known properties of local polynomial estimators. In applications, the most common choices are $p = 1$ for τ_0 (local-linear sharp RD estimator), $p = 2$ for τ_1 (local-quadratic sharp kink RD estimator), $p = 1$ for ς_0 (local-linear fuzzy RD estimator), and $p = 2$ for ς_1 (local-quadratic fuzzy kink RD estimator).

The function `rdrobust()` implements the above RD point estimators with options `c` to set $\bar{x}$, `deriv` to set ν, `p` to set p, `kernel` to set $K(\cdot)$, and `h` to set h_n, among others.

Bandwidth selectors

The main obstacle in the practical implementation of RD local polynomial estimators is bandwidth selection. The package **rdrobust** implements the main two approaches for bandwidth selection available in the literature: (i) plug-in rules based on mean squared error (MSE) expansions, and (ii) cross validation. IK provide a comprehensive review of these approaches.

- **Direct plug-in rules**. Direct plug-in (DPI) approaches to bandwidth selection are based on a mean squared error (MSE) expansion of the sharp RD estimators, leading to the MSE-optimal choice

$$h_{\mathtt{MSE},\nu,p} = C_{\mathtt{MSE},\nu,p}\, n^{-\frac{1}{2p+3}}, \qquad C_{\mathtt{MSE},\nu,p} = \left(\frac{(1+2\nu)\mathsf{V}_{\nu,p}}{2(p+1-\nu)\mathsf{B}^2_{\nu,p}}\right)^{\frac{1}{2p+3}},$$

where $\mathsf{B}_{\nu,p}$ and $\mathsf{V}_{\nu,p}$ are the leading asymptotic bias and variance of the RD estimator, respectively. The package **rdrobust** implements two data-driven versions of this MSE-optimal bandwidth, one proposed by IK (available for $\nu = 0$ only) and the other proposed by CCT (valid for all choices of ν). Both implementations include regularization as originally discussed in IK, although the option `scalereg` allows users to remove it. The IK implementation, denoted by $\hat{h}_{\mathtt{IK},0,p}$, may be viewed as a nonparametric first-generation plug-in rule (e.g., Wand and Jones, 1995), sometimes called a DPI-1 (direct plug-in of order 1) selector. The CCT implementation, denoted by $\hat{h}_{\mathtt{CCT},\nu,p}$, may be viewed as a second-generation plug-in bandwidth selection approach. Further details on implementation of the bandwidth selectors may be found in IK, CCT and their supplemental materials.

- **Cross validation**. This bandwidth choice is implemented as follows:

$$\hat{h}_{\mathtt{CV},p} = \arg\min_{h>0} \mathsf{CV}_\delta(h), \qquad \mathsf{CV}_\delta(h) = \sum_{i=1}^{n} \mathbb{1}(X_{-,[\delta]} \le X_i \le X_{+,[\delta]})\left(Y_i - \hat{\mu}_p(X_i;h)\right)^2,$$

where

$$\hat{\mu}_p(x;h) = \begin{cases} \mathbf{e}_0'\hat{\beta}_{Y+,p}(x,h) & \text{if } x > \bar{x} \\ \mathbf{e}_0'\hat{\beta}_{Y-,p}(x,h) & \text{if } x < \bar{x} \end{cases},$$

and, for $\delta \in (0,1)$, $X_{-,[\delta]}$ and $X_{+,[\delta]}$ denote the δ-th quantile of $\{X_i : X_i < \bar{x}\}$ and $\{X_i : X_i > \bar{x}\}$, respectively. See IK for further discussion on this alternative approach, which is valid only for sharp RD designs ($\nu = 0$).

The function `rdbwselect()` implements the above bandwidth selectors.

Asymptotic properties and confidence intervals

We briefly review the main asymptotic properties of the local polynomial RD estimators, with particular emphasis on the properties of the associated confidence interval estimators. Specifically, we discuss three types of confidence intervals (CI) based on Gaussian approximations: (i) conventional CI (assuming "small" bias), (ii) bias-corrected CI (not necessarily requiring undersmoothing), and (iii) robust bias-corrected CI (not necessarily requiring undersmoothing).

- **Optimal point estimators**. The package **rdrobust** implements the following data-driven RD treatment effect point estimators.

$$\begin{array}{lllll} \text{Sharp RD:} & \hat{\tau}_{\nu,p}(\hat{h}_{\mathtt{CCT},\nu,p}), & \hat{\tau}_{0,p}(\hat{h}_{\mathtt{IK},0,p}), & \hat{\tau}_{0,p}(\hat{h}_{\mathtt{CV},p}), & \nu \le p. \\ \text{Fuzzy RD:} & \hat{\varsigma}_{\nu,p}(\hat{h}_{\mathtt{CCT},\nu,p}), & \hat{\varsigma}_{0,p}(\hat{h}_{\mathtt{IK},0,p}), & \hat{\varsigma}_{0,p}(\hat{h}_{\mathtt{CV},p}), & \nu \le p. \end{array}$$

These estimators are constructed employing MSE-optimal bandwidth choices for the sharp RD case, which means that $\hat{\tau}_{\nu,p}(\hat{h}_{\mathtt{CCT},\nu,p})$, $\hat{\tau}_{0,p}(\hat{h}_{\mathtt{IK},0,p})$ and $\hat{\tau}_{0,p}(\hat{h}_{\mathtt{CV},p})$ may be interpreted as consistent and (asymptotically) MSE-optimal point estimators of τ_ν. For the fuzzy RD cases, the bandwidth choices employed are technically optimal only for the numerator of the estimators, but since the rate of the MSE-optimal bandwidth choice does not differ from the sharp RD case, the estimators $\hat{\varsigma}_{\nu,p}(\hat{h}_{\mathtt{CCT},\nu,p})$, $\hat{\varsigma}_{0,p}(\hat{h}_{\mathtt{IK},\nu,p})$ and $\hat{\varsigma}_{0,p}(\hat{h}_{\mathtt{CV},p})$ may also be viewed as consistent and (asymptotically) MSE-optimal point estimators of ς_ν.

- **Sharp RD confidence intervals**. Confidence intervals accompanying the point estimators discussed above rely on the following distributional approximation:

$$\sqrt{nh_n^{1+2\nu}}\left(\hat{\tau}_{\nu,p}(h_n) - \tau_\nu - h_n^{p+1-\nu}\,\mathsf{B}_{\nu,p}\right) \to_{\mathsf{d}} \mathcal{N}(0,\mathsf{V}_{\nu,p}), \qquad \nu \le p, \tag{1}$$

where $\mathsf{B}_{\nu,p}$ and $\mathsf{V}_{\nu,p}$ denote, respectively, the asymptotic bias and variance of the RD estimator.

Conventional confidence intervals. An asymptotic $100(1-\alpha)$-percent confidence interval for τ_ν is

$$\mathsf{CI}_{1-\alpha}(h_n) = \left[\,\hat{\tau}_{\nu,p}(h_n) \pm \Phi^{-1}_{1-\alpha/2}\sqrt{\frac{\mathsf{V}_{\nu,p}}{nh_n^{1+2\nu}}}\,\right],$$

where Φ_a^{-1} denotes the appropriate quantile of the Gaussian distribution (e.g., 1.96 for $a = .975$). This approach is valid only if the leading bias in (1) is "small". This smoothing bias is ignored by relying on an "undersmoothing" argument, that is, by assuming the bandwidth chosen is "small" enough so that the bias is negligible. In practice, however, this procedure may be difficult to implement because most bandwidth selectors, such as $h_{\mathtt{MSE},\nu,p}$, will not satisfy the conditions required for undersmoothing. This fact implies that most empirical bandwidth selectors could in principle lead to a non-negligible leading bias in the distributional approximation, which in turn will bias the associated confidence intervals.

Bias-corrected confidence intervals. As an alternative to undersmoothing, we can directly bias-correct the estimator by constructing an estimator of $\mathsf{B}_{\nu,p}$, which is then subtracted from the RD point estimate in an attempt to eliminate the leading bias in (1). The resulting asymptotic $100(1-\alpha)$-percent confidence interval for τ_ν is

$$\mathsf{CI}^{\mathsf{bc}}_{1-\alpha}(h_n, b_n) = \left[\left(\hat{\tau}_{\nu,p}(h_n) - h_n^{p+1-\nu}\ \hat{\mathsf{B}}_{\nu,p,q}\right) \pm \Phi^{-1}_{1-\alpha/2}\sqrt{\frac{\mathsf{V}_{\nu,p}}{nh_n^{1+2\nu}}}\right],$$

where $\hat{\mathsf{B}}_{\nu,p,q}$ denotes the bias estimate, which is constructed using a possibly different bandwidth b_n. To implement this approach, CCT propose a MSE-optimal bandwidth choice of b_n for the bias estimator $\hat{\mathsf{B}}_{\nu,p,q}$ taking the form $b_{\mathtt{MSE},\nu,p,q} = \mathsf{C}_{\nu,p,q}^{1/(2q+3)}\ n^{-1/(2q+3)}$, where $\mathsf{C}_{\nu,p,q}$ denotes a constant depending on the data generating process. CCT also discuss an implementation procedure of $b_{\mathtt{MSE},\nu,p,q}$, leading to the data-driven estimator denoted by $\hat{b}_{\mathtt{CCT},\nu,p,q}$.

Robust bias-corrected confidence intervals. The confidence intervals discussed so far have some unappealing properties that may affect their performance in applications. On the one hand, the confidence intervals $\mathsf{CI}_{1-\alpha}(h_n)$ require undersmoothing (or, alternatively, a "small" bias), which may lead to coverage distortions in cases where the bias is important. On the other hand, the bias-corrected confidence intervals $\mathsf{CI}^{\mathsf{bc}}_{1-\alpha}(h_n, b_n)$, while theoretically justified for a larger range of bandwidths, are usually regarded as having poor performance in empirical settings, also leading to potentially large coverage distortions in applications.

CCT propose an alternative, more robust confidence interval formula based on the bias-corrected RD treatment effect estimators, but employing a different variance for studentization purposes. Intuitively, the bias-corrected RD estimator does not perform well in finite-samples because the bias estimate introduces additional variability in the statistic, which is not accounted for when forming the associated confidence intervals $\mathsf{CI}^{\mathsf{bc}}_{1-\alpha}(h_n, b_n)$. Thus, CCT propose the asymptotic $100(1-\alpha)$-percent confidence interval for τ_ν given by

$$\mathsf{CI}^{\mathsf{rbc}}_{1-\alpha}(h_n, b_n) = \left[\left(\hat{\tau}_{\nu,p}(h_n) - h_n^{p+1-\nu}\ \hat{\mathsf{B}}_{\nu,p,q}\right) \pm \Phi^{-1}_{1-\alpha/2}\sqrt{\mathsf{V}^{\mathsf{bc}}_{n,\nu,p,q}}\right],$$

which includes the alternative variance formula $\mathsf{V}^{\mathsf{bc}}_{n,\nu,p,q}(h_n, b_n)$ accounting for the additional variability introduced by the bias estimate. See CCT and Calonico, Cattaneo, and Farrell (2015a) for further details.

- **Sharp RD variance estimation**. To construct fully feasible confidence intervals the unknown asymptotic variance is replaced by a consistent estimator thereof. The asymptotic variance formulas introduced above have a "sandwich" structure coming from the weighted least-squares structure of local polynomials. The package **rdrobust** offers two distinct valid variance estimators, employing either "plug-in estimated residuals" or "fixed-matches estimated residuals".

  Plug-in estimated residuals. In this approach, the unknown residuals are estimated directly using the RD local polynomial estimator. Usually the same bandwidth h_n is employed, although his choice may not be optimal and could lead to poor finite-sample performance of the variance estimator. $\check{\mathsf{V}}_{n,\nu,p}$ and $\check{\mathsf{V}}^{\mathsf{bc}}_{n,\nu,p,q}$ denote the resulting estimators of $\mathsf{V}_{n,\nu,p}$ and $\mathsf{V}^{\mathsf{bc}}_{n,\nu,p,q}$, respectively.

  Fixed-matches estimated residuals. CCT propose an alternative variance estimator employing a different construction for the residuals, motivated by the work of Abadie and Imbens (2006). This estimator is constructed using a simple nearest-neighbor (or fixed-matches) estimator for the residuals. $\hat{\mathsf{V}}_{n,\nu,p}$ and $\hat{\mathsf{V}}^{\mathsf{bc}}_{n,\nu,p,q}$ denote the resulting estimators of $\mathsf{V}_{n,\nu,p}$ and $\mathsf{V}^{\mathsf{bc}}_{n,\nu,p,q}$, respectively.

- **Extensions to Fuzzy RD confidence intervals**. All the ideas and results presented above extend to the case of fuzzy RD designs, which we omit here to conserve space. See IK and CCT for further details. The package **rdrobust** also implements fuzzy RD estimators and confidence intervals.

The function `rdrobust()` may be used to conduct inference in all RD settings; by default, this function employs the function `rdbwselect()` for bandwidth selection. Specifically, assuming y is the output variable, t is the treatment status variable, x is the running variable and the cutoff is $c = 0$, we have the following generic cases (with default options and bandwidth selection procedure):

- Sharp RD:
  `rdrobust(y = y,x = x)`
- Sharp Kink RD:
  `rdrobust(y = y,x = x,deriv = 1)`
- Fuzzy RD:
  `rdrobust(y = y,x = x,fuzzy = t)`
- Fuzzy Kink RD:
  `rdrobust(y = y,x = x,fuzzy = t,deriv = 1)`

RD plots

Because of the simplicity of the RD design, it is customary (and advisable) to summarize the main features of the RD design in a graphical way. The package **rdrobust** also implements the results in Calonico, Cattaneo, and Titiunik (in press), which offer several optimal data-driven choices of tuning parameters useful to produce several versions of the popular RD plots. These plots present global and local estimates of the regression functions, $\mu_{Y-}(x)$ and $\mu_{Y+}(x)$, in an attempt to describe their shapes for control ($X_i < \bar{x}$) and treated ($X_i \geq \bar{x}$) units. The plot typically includes two smooth "global" polynomial regression curve estimates, for control and treatment units separately, as well as binned sample means of the outcome, which are included either to capture the local behavior of the underlying regression functions or to represent the overall variability of the raw data in a disciplined way. These binned sample means are used to either (i) investigate whether other discontinuities are present, and (ii) depict an informative "cloud of points" around the smooth global polynomial fits.

The first ingredient of the RD plot (two regression curves estimated for $X_i < \bar{x}$ and $X_i \geq \bar{x}$ separately) is easy to construct because it requires only estimating a polynomial regression on the data; typical choices are 4-th and 5-th order global polynomials. The second ingredient of the RD plot requires computing sample means over non-overlapping regions of the support of the running variable X_i for control and treatment units, separately. For implementation the researcher needs to choose the number of bins to be used, denoted by $J_{-,n}$ (control) and $J_{+,n}$ (treatment), and the length of each bin; evenly-spaced bins is the most common partitioning scheme used. Calonico, Cattaneo, and Titiunik (in press) study the problem of selecting an optimal number of bins $J_{-,n}$ and $J_{+,n}$ under two partitioning schemes, evenly-spaced (ES) and quantile-spaced (QS), and propose several consistent nonparametric selectors using spacings and polynomial regression estimators. In particular, they propose two approaches to select the number of bins: (i) IMSE-optimal (tailored to approximate the underlying regression functions), and (ii) Mimicking Variance (tailored to represent the overall variability of the data in a disciplined way). These two proposed choices take the form, respectively,

$$\text{IMSE-optimal}: \quad J^*_{-,n} = \left\lceil \mathscr{C}_- \; n^{1/3} \right\rceil \quad \text{and} \quad J^*_{+,n} = \left\lceil \mathscr{C}_+ \; n^{1/3} \right\rceil,$$

and

$$\text{Mimicking Variance}: \quad J^*_{-,n} = \left\lceil \mathscr{C}_- \; \frac{n}{\log(n)^2} \right\rceil \quad \text{and} \quad J^*_{+,n} = \left\lceil \mathscr{C}_+ \; \frac{n}{\log(n)^2} \right\rceil,$$

where $\lceil \cdot \rceil$ denotes the ceiling function, and the unknown constants take different values depending on the targeted method and partitioning scheme used. Once the partitioning scheme is selected, the optimal choices $J^*_{-,n}$ and $J^*_{+,n}$ can be estimated using preliminary plug-in estimators of $\mathscr{C}_-$ and $\mathscr{C}_+$.

The function `rdplot()` offers eight distinct automatic implementations for RD plots depending on (i) the choice of partitioning (ES or QS), (ii) the goal of the plot (IMSE-optimal or Mimicking Variance), and (iii) the estimation approach used (spacings or polynomial regression). Specifically, the function `rdplot()` covers the following.

- Population quantities:
  $J_{\texttt{ES}\text{-}\mu,\cdot,n}$ = IMSE-optimal choice with evenly-spaced (ES) bins.
  $J_{\texttt{ES}\text{-}\vartheta,\cdot,n}$ = Mimicking Variance choice with evenly-spaced (ES) bins.
  $J_{\texttt{QS}\text{-}\mu,\cdot,n}$ = IMSE-optimal choice with quantile-spaced (QS) bins.
  $J_{\texttt{QS}\text{-}\vartheta,\cdot,n}$ = Mimicking Variance choice with quantile-spaced (QS) bins.

- Estimators:
  $\hat{J}_{\texttt{ES}\text{-}\mu,\cdot,n}$ and $\hat{J}_{\texttt{QS}\text{-}\mu,\cdot,n}$ = spacings implementations of $J_{\texttt{ES}\text{-}\mu,\cdot,n}$ and $J_{\texttt{QS}\text{-}\mu,\cdot,n}$, respectively.
  $\check{J}_{\texttt{ES}\text{-}\mu,\cdot,n}$ and $\check{J}_{\texttt{QS}\text{-}\mu,\cdot,n}$ = polynomial regression implementations of $J_{\texttt{ES}\text{-}\mu,\cdot,n}$ and $J_{\texttt{QS}\text{-}\mu,\cdot,n}$, respectively.
  $\hat{J}_{\texttt{ES}\text{-}\vartheta,\cdot,n}$ and $\hat{J}_{\texttt{QS}\text{-}\vartheta,\cdot,n}$ = spacings implementations of $J_{\texttt{ES}\text{-}\vartheta,\cdot,n}$ and $J_{\texttt{QS}\text{-}\vartheta,\cdot,n}$, respectively.
  $\check{J}_{\texttt{ES}\text{-}\vartheta,\cdot,n}$ and $\check{J}_{\texttt{QS}\text{-}\vartheta,\cdot,n}$ = polynomial regression implementations of $J_{\texttt{ES}\text{-}\vartheta,\cdot,n}$ and $J_{\texttt{QS}\text{-}\vartheta,\cdot,n}$, respectively.

Further details on implementation and syntax are given in the help file of the function `rdplot()`. For other technical and methodological details see Calonico, Cattaneo, and Titiunik (in press).

The rdrobust package: Empirical illustration

We employ an extract of the dataset constructed by Cattaneo, Frandsen, and Titiunik (2015) to illustrate some of the features of our R package **rdrobust**. This dataset contains information on elections for the U.S. Senate during the period 1914–2010. We focus on the RD effect of the Democratic party winning a U.S. Senate seat on the vote share obtained in the following election for that same seat, mimicking the analysis conducted in Lee (2008) for the U.S. House. The dataset `rdrobust_RDsenate` contains two variables: *vote* and *margin*. The variable *vote* records the state-level vote share of the Democratic party in a given statewide election for a Senate seat, while the variable *margin* records the margin of victory of the Democratic party in the previous election for the same Senate seat (i.e., six years prior).

First, we load the database and present basic summary statistics. The functions included in the R package **rdrobust** allow for missing values, which are automatically excluded for estimation purposes.

```
> library(rdrobust)
> data(rdrobust_RDsenate)
> vote <- rdrobust_RDsenate$vote
> margin <- rdrobust_RDsenate$margin
> summary(vote)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00   42.67   50.55   52.67   61.35  100.00      93
> summary(margin)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-100.000  -12.210    2.166    7.171   22.770  100.000
```

This data set has a total of 1,297 complete observations. The variable *margin* ranges from −100 to 100, and records the margin of victory in a given election for a given U.S. Senate seat, defined as the vote share of the Democratic party minus the vote share of the strongest opponent. When *margin* is above zero, the Democratic party wins the election for that seat, otherwise it loses. The variable *vote* ranges from 0 to 100 because it records the outcome of the (two-periods ahead) election for that given seat. Thus, observations for years 2008 and 2010 have missing *vote*. As it is usual in the literature, we exploit the discontinuity in incumbency status that occurs at 0 on *margin* to employ an RD design.

We use `rdplot()` to construct an automatic plot of the RD design.

```
> (rdplot(y = vote, x = margin, title = "RD Plot - Senate Elections Data",
+   y.label = "Vote Share in Election at time t+1",
+   x.label = "Vote Share in Election at time t"))
Call:
rdplot(y = vote, x = margin, title = "RD Plot - Senate Elections Data",
   x.label = "Vote Share in Election at time t",
   y.label = "Vote Share in Election at time t+1")

Method: mimicking variance evenly-spaced method using spacings estimators


                          Left   Right
Number of Obs.            595    702
Polynomial Order          4      4
Scale                     1      1

Selected Bins             15     35
Bin Length                6.6614 2.8561

IMSE-optimal bins         8      9
Mimicking Variance bins   15     35

Relative to IMSE-optimal:
Implied scale             1.8750 3.8889
WIMSE variance weight     0.1317 0.0167
WIMSE bias weight         0.8683 0.9833
```

Figure 1 is constructed using the default options in the command `rdplot`, which produce an RD plot with evenly-spaced bins selected to mimic the underlying variability of the data and is

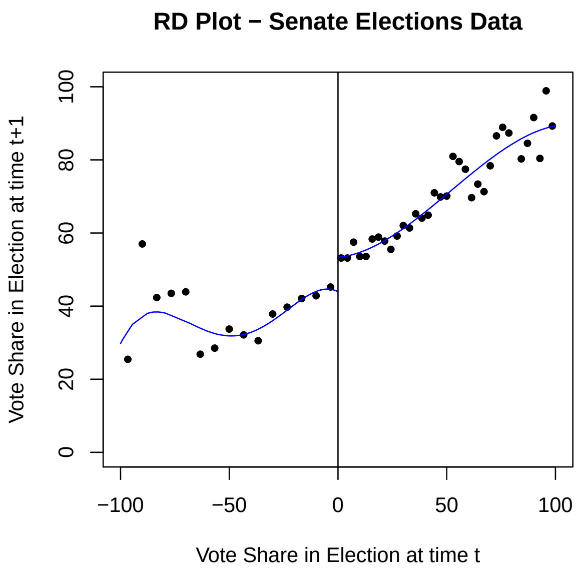


Figure 1: Automatic RD plot with evenly spaced bins (Mimicking Variance).

implemented using spacings estimators. Using the notation introduced above, the number of optimal bins for control and treatment units are $\hat{J}_{-,n} = 15$ and $\hat{J}_{+,n} = 35$, respectively, implying bin lengths of 6.661 and 2.856 percentage points, respectively. The global polynomial is constructed using a $4th$ degree polynomial ($p = 4$ for $\hat{\mu}_{-.p,1}(x)$ and $\hat{\mu}_{+.p,1}(x)$).

Next, we construct an alternative RD plot using evenly-spaced bins selected to trace out the underlying regression function (i.e., using the IMSE-optimal selector), also implemented using spacings estimators. The resulting plot is given in Figure 2.

```
> (rdplot(y = vote, x = margin, binselect = "es",
+   title = "RD Plot - Senate Elections Data",
+   y.label = "Vote Share in Election at time t+1",
+   x.label = "Vote Share in Election at time t"))
Call:
rdplot(y = vote, x = margin, binselect = "es", x
   title = "RD Plot - Senate Elections Data",
   x.label = "Vote Share in Election at time t",
   y.label = "Vote Share in Election at time t+1")

Method: IMSE-optimal evenly-spaced method using spacings estimators


                          Left    Right
Number of Obs.            595     702
Polynomial Order          4       4
Scale                     1       1

Selected Bins             8       9
Bin Length                12.4901 11.1071

IMSE-optimal bins         8       9
Mimicking Variance bins   15      35

Relative to IMSE-optimal:
Implied scale             1.0000  1.0000
WIMSE variance weight     0.5000  0.5000
WIMSE bias weight         0.5000  0.5000
```

While providing a good approximation to the underlying regression function (taking the global polynomial fit as benchmark), the IMSE-optimal number of bins will usually be too small in applications. This happens because the optimal formulas seek to balance squared bias and variance in order to approximate the underlying regression function globally. To obtain a visual "cloud of points" we need to increase the number of bins, that is, undersmooth the estimator. In other words, in order to

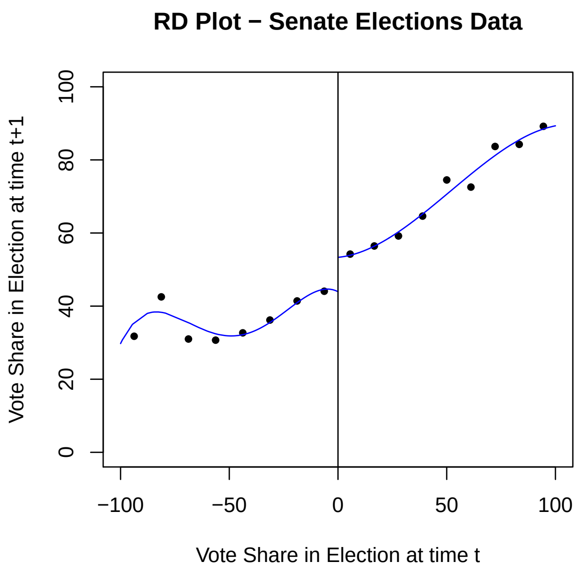


Figure 2: Automatic RD plot with evenly spaced bins (IMSE-optimal).

increase the overall variability of the plotted points, we may reduce the bin-length – which is done by increasing the total number of bins used. This may be easily done using the option scale as follows:

```
> (rdplot(y = vote, x = margin, binselect = "es", scale = 5,
+   title = "RD Plot - Senate Elections Data",
+   y.label = "Vote Share in Election at time t+1",
+   x.label = "Vote Share in Election at time t"))
Call:
rdplot(y = vote, x = margin, binselect = "es", scale = 5,
   title = "RD Plot - Senate Elections Data",
   x.label = "Vote Share in Election at time t",
   y.label = "Vote Share in Election at time t+1")

Method: IMSE-optimal evenly-spaced method using spacings estimators


                          Left   Right
Number of Obs.            595    702
Polynomial Order          4      4
Scale                     5      5

Selected Bins             40     45
Bin Length                2.4980 2.2214

IMSE-optimal bins         8      9
Mimicking Variance bins   15     35

Relative to IMSE-optimal:
Implied scale             5.0000 5.0000
WIMSE variance weight     0.0079 0.0079
WIMSE bias weight         0.9921 0.9921
```

Figure 3 shows the resulting (undersmoothed) RD plot, where now the number of bins used is five times larger than the optimal choice in an integrated mean squared error sense. The resulting estimator is naturally more variable than before.

Next, we conduct fully data-driven RD treatment effect estimation and inference. The function rdrobust() using its default options leads to the following output:

```
> rdrobust(y = vote, x = margin)

Call:
rdrobust(y = vote, x = margin)
```

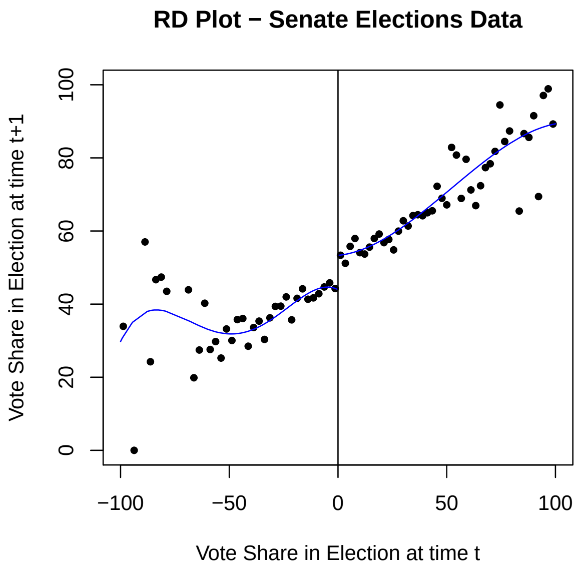


Figure 3: Automatic RD plot with evenly spaced bins and scaled down bin length (IMSE-optimal).

```
Summary:

Number of Obs 1297
NN Matches    3
BW Type       CCT
Kernel Type   Triangular

                   Left    Right
Number of Obs      343     310
Order Loc Poly (p) 1       1
Order Bias (q)     2       2
BW Loc Poly (h)    16.7936 16.7936
BW Bias (b)        27.4372 27.4372
rho (h/b)          0.6121  0.6121

Estimates:
             Coef   Std. Err. z      P>|z|  CI Lower CI Upper
Conventional 7.4253 1.4954    4.9656 0.0000 4.4944   10.3562
Robust                               0.0000 4.0697   10.9833
```

These results contain a variety of information, which is organized in three panels. The first two contain a summary of the main choices selected to construct the RD treatment effect estimators, while the lower panel includes the main estimation results. Specifically, using the notation introduced above, this table shows:

1. The total number of observations is $1,297$, with effective 343 control and 310 treated units (given the bandwidth h_n chosen; see below). The estimation is conducted using a local-linear ($p = 1$) estimator with a local-quadratic ($q = 2$) bias-correction estimate, with a triangular kernel. The standard error estimators are the ones proposed by CCT, computed using 3 nearest-neighbors.

2. The bandwidth selection procedure is the one proposed by CCT, leading to $\hat{h}_{\texttt{CCT},\nu,p} = 16.7936$ with $p = 1$, and $\hat{b}_{\texttt{CCT},\nu,p,q} = 27.4372$ with $q = 2$. Recall $\hat{h}_{\texttt{CCT},\nu,p}$ and $\hat{b}_{\texttt{CCT},\nu,p,q}$ are the estimated bandwidths used to construct the RD point estimator and the RD bias-correction, respectively, and $\nu = 0$ in this illustration.

3. The RD point estimator is $\hat{\tau}_p(\hat{h}_{\texttt{CCT},\nu,p}) = 7.4253$ and the RD robust confidence intervals is $\hat{\mathsf{CI}}^{rbc}_{1-\alpha}(\hat{h}_{\texttt{CCT},\nu,p}, \hat{b}_{\texttt{CCT},\nu,p,q}) = [\ 4.0697\ ,\ 10.9833\ ]$, with a default choice of $\alpha = 0.05$.

The function `rdrobust()` also offers a more detailed output, which includes all the point estimators, standard errors and confidence intervals discussed previously. These results are retrieved using the `all = TRUE` option. The corresponding output is as follows:

```
> rdrobust(y = vote, x = margin, all = TRUE)

Call:
rdrobust(y = vote, x = margin, all = TRUE)

Summary:

Number of Obs 1297
NN Matches    3
BW Type       CCT
Kernel Type   Triangular

                   Left    Right
Number of Obs      343     310
Order Loc Poly (p) 1       1
Order Bias (q)     2       2
BW Loc Poly (h)    16.7936 16.7936
BW Bias (b)        27.4372 27.4372
rho (h/b)          0.6121  0.6121

Estimates:
               Coef   Std. Err. z      P>|z|  CI Lower CI Upper
Conventional   7.4253 1.4954    4.9656 0.0000 4.4944   10.3562
Bias-Corrected 7.5265 1.4954    5.0333 0.0000 4.5957   10.4574
Robust         7.5265 1.7637    4.2675 0.0000 4.0697   10.9833
```

Finally, we illustrate all the bandwidth selection procedures contained in our package. To this end, we employ our companion function `rdbwselect()` to compare the CCT bandwidth selectors with the IK and CV approaches. We have:

```
> rdbwselect(y = vote, x = margin, all = TRUE)

Call:
rdbwselect(y = vote, x = margin, all = TRUE)

BW Selector   All
Number of Obs 1297
NN Matches    3
Kernel Type   Triangular

                   Left Right
Number of Obs      595  702
Order Loc Poly (p) 1    1
Order Bias (q)     2    2

           h        b
CCT 16.79357 27.43722
IK  15.66761 16.48524
CV  35.42113       NA
```

In this case we employed the option `all = TRUE`, which computes the three bandwidth selectors briefly discussed above. Notice that the option `CV` is currently not available for derivative estimation. To further understand the performance of the CV approach, we include a graph of the CV objective function over the grid being considered. This is done using the option `cvplot` as shown next (in this example we also changed the grid features to obtain a better plot, and to show this additional functionality in action as well).

```
> rdbwselect(y = vote, x = margin, bwselect = "CV",
+   cvgrid_min = 10, cvgrid_max = 80, cvplot = TRUE)

Call:
rdbwselect(y = vote, x = margin, bwselect = "CV", cvgrid_min = 10,
    cvgrid_max = 80, cvplot = TRUE)

BW Selector   CV
Number of Obs 1297
```

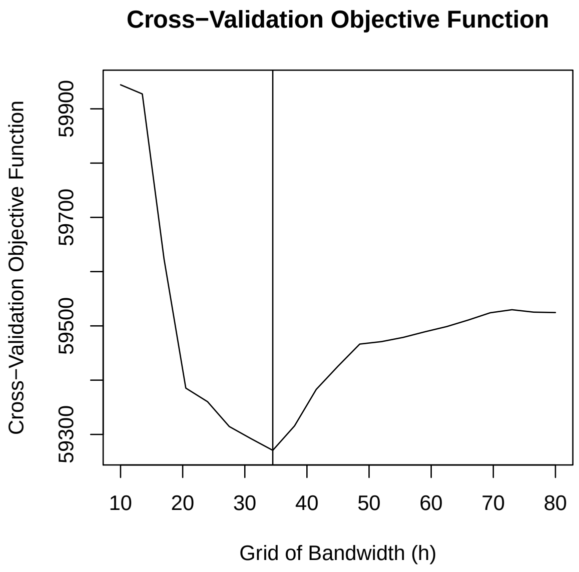


Figure 4: Values of the CV function over the selected grid.

```
NN Matches    3
Kernel Type   Triangular

                    Left Right
Number of Obs       595  702
Order Loc Poly (p)  1    1
Order Bias (q)      2    2

     h    b
   34.5 34.5
```

Figure 4 shows the CV objective function as a function of a grid of bandwidth. In this example, the cross validation approach delivers a minimum at $\hat{h}_{\texttt{CV},p} = 34.5$.

Our functions contained in the R package **rdrobust** have many other options. For instance, for the main function `rdrobust()` we have the following additional examples (output is not provided to conserve space).

- Estimation using uniform kernel:
  `rdrobust(y = vote,x = margin,kernel = "uniform")`
- Estimation using the IK bandwidth selector:
  `rdrobust(y = vote,x = margin,bwselect = "IK")`
- Estimation using the CV bandwidth selector:
  `rdrobust(y = vote,x = margin,bwselect = "CV")`
- Estimation using $h_n = 15$ and $\rho = h_n/b_n = 0.8$:
  `rdrobust(y = vote,x = margin,h = 15,rho = 0.8)`
- Estimation using $p = 2$ and $q = 4$:
  `rdrobust(y = vote,x = margin,p = 2,q = 4)`
- Estimation using plug-in residuals estimates:
  `rdrobust(y = vote,x = margin,vce = "resid")`

Conclusions

We introduced the main features of the R package **rdrobust**, which includes the functions `rdrobust()`, `rdbwselect()`, and `rdplot()` designed to conduct data-driven nonparametric robust inference in RD designs. This implementation covers average RD treatment effects at the cutoff in the sharp RD, sharp kink RD, fuzzy RD and fuzzy kink RD designs. A full description of this package may be found in its manual and help files. A companion Stata package offering the same structure and capabilities is described in Calonico, Cattaneo, and Titiunik (2014b).

Acknowledgments

We thank the Editor Deepayan Sarkar, two anonymous reviewers, Richard Anderson, Devin Caughey and Marko Klašnja for comments and suggestions on previous versions of our R package **rdrobust**. The authors gratefully acknowledge financial support from the National Science Foundation (SES 1357561).

Bibliography

A. Abadie and G. W. Imbens. Large sample properties of matching estimators for average treatment effects. *Econometrica*, 74(1):235–267, 2006. [p42]

S. Calonico, M. D. Cattaneo, and R. Titiunik. Robust nonparametric confidence intervals for regression-discontinuity designs. *Econometrica*, 82(6):2295–2326, 2014a. [p38]

S. Calonico, M. D. Cattaneo, and R. Titiunik. Robust data-driven inference in the regression-discontinuity design. *Stata Journal*, 14(4):909–946, 2014b. [p38, 49]

S. Calonico, M. D. Cattaneo, and M. H. Farrell. On the effect of bias estimation on coverage accuracy in nonparametric estimation. Working paper, University of Michigan, 2015a. [p42]

S. Calonico, M. D. Cattaneo, and R. Titiunik. *rdrobust: Robust Data-Driven Statistical Inference in Regression-Discontinuity Designs*, 2015b. URL `http://CRAN.R-project.org/package=rdrobust`. R package version 0.80. [p38]

S. Calonico, M. D. Cattaneo, and R. Titiunik. Optimal data-driven regression discontinuity plots. *Journal of the American Statistical Association*, in press. doi: 10.1080/01621459.2015.1017578. [p38, 43, 44]

D. Card, D. S. Lee, Z. Pei, and A. Weber. Inference on causal effects in a generalized regression kink design. Working paper, UC Berkley, 2014. [p39]

M. D. Cattaneo and M. H. Farrell. Optimal convergence rates, Bahadur representation, and asymptotic normality of partitioning estimators. *Journal of Econometrics*, 174(2):127–143, 2013. [p38]

M. D. Cattaneo, B. Frandsen, and R. Titiunik. Randomization inference in the regression discontinuity design: An application to party advantages in the U.S. Senate. *Journal of Causal Inference*, 3(1):1–24, 2015. [p44]

J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications*. Chapman & Hall/CRC, New York, 1996. [p40]

J. Hahn, P. Todd, and W. van der Klaauw. Identification and estimation of treatment effects with a regression-discontinuity design. *Econometrica*, 69(1):201–209, 2001. [p39]

J. J. Heckman and E. J. Vytlacil. Econometric evaluation of social programs, part I: Causal models, structural models and econometric policy evaluation. In J. Heckman and E. Leamer, editors, *Handbook of Econometrics*, volume VI, pages 4780–4874. Elsevier Science B.V., 2007. [p39]

G. Imbens and T. Lemieux. Regression discontinuity designs: A guide to practice. *Journal of Econometrics*, 142(2):615–635, 2008. [p38]

G. W. Imbens and K. Kalyanaraman. Optimal bandwidth choice for the regression discontinuity estimator. *Review of Economic Studies*, 79(3):933–959, 2012. [p38]

G. W. Imbens and J. M. Wooldridge. Recent developments in the econometrics of program evaluation. *Journal of Economic Literature*, 47(1):5–86, 2009. [p39]

D. S. Lee. Randomized experiments from non-random selection in U.S. House elections. *Journal of Econometrics*, 142(2):675–697, 2008. [p39, 44]

D. S. Lee and T. Lemieux. Regression discontinuity designs in economics. *Journal of Economic Literature*, 48(2):281–355, 2010. [p38]

J. Porter. Estimation in the regression discontinuity model. Working paper, University of Wisconsin, 2003. [p39]

StataCorp. *Stata Data Analysis Statistical Software: Release 13*. StataCorp LP, College Station, TX, 2013. URL `http://www.stata.com/`. [p38]

M. Wand and M. Jones. *Kernel Smoothing*. Chapman & Hall/CRC, Florida, 1995. [p41]

Sebastian Calonico
Department of Economics
University of Miami
Coral Gables, Florida
USA
scalonico@bus.miami.edu

Matias D. Cattaneo
Department of Economics
University of Michigan
Ann Arbor, Michigan
USA
cattaneo@umich.edu

Rocío Titiunik
Department of Political Science
University of Michigan
Ann Arbor, Michigan
USA
titiunik@umich.edu

Frames2: A Package for Estimation in Dual Frame Surveys

by Antonio Arcos, David Molina, Maria Giovanna Ranalli and María del Mar Rueda

Abstract Data from complex survey designs require special consideration with regard to estimation of finite population parameters and corresponding variance estimation procedures, as a consequence of significant departures from the simple random sampling assumption. In the past decade a number of statistical software packages have been developed to facilitate the analysis of complex survey data. All these statistical software packages are able to treat samples selected from one sampling frame containing all population units. Dual frame surveys are very useful when it is not possible to guarantee a complete coverage of the target population and may result in considerable cost savings over a single frame design with comparable precision. There are several estimators available in the statistical literature but no existing software covers dual frame estimation procedures. This gap is now filled by package **Frames2**. In this paper we highlight the main features of the package. The package includes the main estimators in dual frame surveys and also provides interval confidence estimation.

Introduction

Classic sampling theory usually assumes the existence of one sampling frame containing all finite population units. Then, a probability sample is drawn according to a given sampling design and information collected is used for estimation and inference purposes. In traditional 'design-based' inference the population data are regarded as fixed and the randomness comes entirely from the sampling procedure. The most used design-based estimator is the Horvitz-Thompson estimator that is unbiased for the population total if the sampling frame includes all population units, if all sampled units respond and if there is no measurement error. In the presence of auxiliary information, there exist several procedures to obtain more efficient estimators for population means and totals of variables of interest; in particular, customary ratio, regression, raking, post-stratified and calibration estimators. Several software packages have been developed to facilitate the analysis of complex survey data and implement some of these estimators as SAS (SAS Institute Inc., 2013), SPSS (IBM Corporation, 2013), Systat (Systat Software Inc., 2009), Stata (Stata Corporation, 2015), SUDAAN (Research Triangle Institute, 2013) and PCCarp (Fuller et al., 1989). CRAN hosts several R packages that include these design-based methods typically used in survey methodology to treat samples selected from one sampling frame (e.g. **survey**, Lumley 2014; **sampling**, Tillé and Matei 2012; **laeken**, Alfons et al. 2014 or **TeachingSampling**, Gutierrez Rojas 2014, among others). Templ (2014) provides a detailed list of packages that includes methods to analyse complex surveys.

In practice, the assumption that the sampling frame contains all population units is rarely met. Often, one finds that sampling from a frame which is known to cover approximately all units in the population is quite expensive while other frames (e.g. special lists of units) are available for cheaper sampling methods. However, the latter usually only cover an unknown or only approximately known fraction of the population. A common example of frame undercoverage is provided by telephone surveys. Estimation could be affected by serious bias due to the lack of a telephone in some households and the generalised use of mobile phones, which are sometimes replacing fixed (land) lines entirely. The potential for coverage error as a result of the exponential growth of the cell-phone only population has led to the development of dual-frame surveys. In these designs, a traditional sample from the landline frame is supplemented with an independent sample from a register of cell-phone numbers.

The dual frame sampling approach assumes that two frames are available for sampling and that, overall, they cover the entire target population. The most common situation is the one represented in Figure 1 where the two frames, say frame A and frame B, show a certain degree of overlapping, so it is possible to distinguish three disjoint non-empty domains: domain a, containing units belonging to frame A but not to frame B; domain b, containing units belonging to frame B but not to frame A and domain ab, containing units belonging to both frames. As an example, consider a telephone survey where both landline and cell phone lists are available; let A be the landline frame and B the cell phone frame. Then, it is possible to distinguish three types of individuals: landline only units, cell-only units and units with both landline and cell phone, which will compose domain a, b and ab, respectively.

Nevertheless, one can face some other situations depending on the relative positions of the frames. For example, Figure 2 shows the case in which frame B is totally included in frame A, that is, frame B is a subset of frame A. Here domain b is empty. We also may find scenarios where the two sampling frames exactly match, as depicted in Figure 3, where ab is the only non-empty domain. Finally, the scenario where domain ab is empty has no interest from a dual frame perspective, since it can be

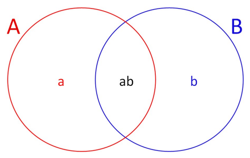


Figure 1: Two frames with overlapping.

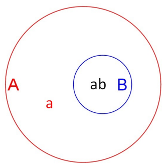


Figure 2: Frame B is included in frame A.

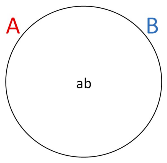


Figure 3: Frame A and frame B exactly match.

considered as a special case of stratified sampling.

Whatever the scenario, an appropriate choice of the frames results in a better coverage of the target population, which, in turn, leads to a better efficiency of estimators calculated from data from dual frame surveys. This point is particularly important when estimating parameters in rare or elusive populations, where undercoverage errors are usually due to the difficulty of finding individuals showing the characteristic under study when sampling from only one general frame. This issue can be dealt with by incorporating a second frame with a high density of members of the rare population so that the two frames are, together, now complete. Dual frame sampling as a method of improvement of efficiency may seem expensive and unviable, but it is not. In fact, Hartley (1962) notes that dual frame surveys can result in important cost savings in comparison with single frame ones with a comparable efficiency. As an additional interesting characteristic, dual frame methodology offers the researcher the possibility to consider different data collection procedures and/or different sampling designs, one for each frame. Dual frame surveys have gained much attention and became largely used by statistical agencies and private organizations to take advantage of these benefits.

Standard software packages for complex surveys cannot be used directly when the sample is obtained from a dual frame survey because the classical design-based estimators are severely biased and there is a underestimation of standard errors. Weighted analyses with standard statistical software, with certain modified weights, can yield correct point estimates of population parameters but still yield incorrect results for estimated standard errors. A number of authors have developed methods for estimating population means and totals from dual (or, more generally, multiple) frame surveys but most of these methods require ad-hoc software for their implementation. To the best of our knowledge, there is no software incorporating these estimation procedures for handling dual frame surveys.

Frames2 (Arcos et al., 2015) tries to fill this gap by providing functions for point and interval estimation from dual frame surveys. The paper is organized as follows. In the next section, we provide an overview of the main point estimators proposed so far in the dual frame context and review also jackknife variance estimation as a tool to compare efficiency for all of them. Subsequently, we present package **Frames2**, discussing guidelines that have been followed to construct it and presenting its principal functions and functionalities. We also provide examples to illustrate how the package works.

Estimation in dual frame surveys

Consider again the situation depicted in Figure 1. Assume we have a finite set of N population units identified by integers, $\mathcal{U} = \{1, \ldots, k, \ldots, N\}$, and let A and B be two sampling frames, both can be incomplete, but it is assumed that together they cover the entire finite population.

Let $\mathcal{A}$ be the set of population units in frame A and $\mathcal{B}$ the set of population units in frame B. The population of interest, $\mathcal{U}$, may be divided into three mutually exclusive domains, $a = \mathcal{A} \cap \mathcal{B}^c, b = \mathcal{A}^c \cap \mathcal{B}$ and $ab = \mathcal{A} \cap \mathcal{B}$. Let N, N_A, N_B, N_a, N_b and N_{ab} be the number of population units in $\mathcal{U}, \mathcal{A}, \mathcal{B}, a, b, ab$, respectively.

Let y be a variable of interest in the population and let y_k be its value on unit k, for $k = 1, \ldots, N$. The objective is to estimate the finite population total $Y = \sum_k y_k$ that can be written as

$$Y = Y_a + Y_{ab} + Y_b,$$

where $Y_a = \sum_{k \in a} y_k, Y_{ab} = \sum_{k \in ab} y_k$ and $Y_b = \sum_{k \in b} y_k$. To this end, independent samples s_A and s_B are drawn from frame A and frame B of sizes n_A and n_B, respectively. Unit k in $\mathcal{A}$ has first-order inclusion probability $\pi_k^A = Pr(k \in s_A)$ and unit k in $\mathcal{B}$ has first-order inclusion probability $\pi_k^B = Pr(k \in s_B)$.

From data collected in s_A, it is possible to compute one unbiased estimator of the total for each domain in frame A, $\hat{Y}_a$ and $\hat{Y}_{ab}^A$, as described below:

$$\hat{Y}_a = \sum_{k \in s_A} \delta_k(a) d_k^A y_k, \qquad \hat{Y}_{ab}^A = \sum_{k \in s_A} \delta_k(ab) d_k^A y_k,$$

where $\delta_k(a) = 1$ if $k \in a$ and 0 otherwise, $\delta_k(ab) = 1$ if $k \in ab$ and 0 otherwise and d_k^A are the weights under the sampling design used in frame A, defined as the inverse of the first order inclusion probabilities, $d_k^A = 1/\pi_k^A$. Similarly, using information included in s_B, one can obtain an unbiased estimator of the total for domain b and another one for domain ab, $\hat{Y}_b$ and $\hat{Y}_{ab}^B$, which can be expressed as

$$\hat{Y}_b = \sum_{k \in s_B} \delta_k(b) d_k^B y_k, \qquad \hat{Y}_{ab}^B = \sum_{k \in s_B} \delta_k(ab) d_k^B y_k,$$

with $\delta_k(b) = 1$ if $k \in b$ and 0 otherwise, and d_k^B the weights under the sampling design used in frame B defined as the inverse of the first order inclusion probabilities, $d_k^B = 1/\pi_k^B$.

Different approaches for estimating the population total from dual frame surveys have been proposed in the literature. Hartley (1962) suggests the use of a parameter, θ, to weight $\hat{Y}_{ab}^A$ and $\hat{Y}_{ab}^B$, providing the estimator

$$\hat{Y}_H = \hat{Y}_a + \theta \hat{Y}_{ab}^A + (1-\theta)\hat{Y}_{ab}^B + \hat{Y}_b, \tag{1}$$

where $\theta \in [0,1]$. Hartley (1974) himself proved that

$$\theta_{opt} = \frac{V(\hat{Y}_{ab}^B) + Cov(\hat{Y}_b, \hat{Y}_{ab}^B) - Cov(\hat{Y}_a, \hat{Y}_{ab}^A)}{V(\hat{Y}_{ab}^A) + V(\hat{Y}_{ab}^B)}$$

is the optimum value for θ so that variance of the estimator with respect to the design is minimized. In practice, θ_{opt} cannot be computed, since population variances and covariances involved in its calculation are unknown, so they must be estimated from sampling data. An estimator for the variance of $\hat{Y}_H$ can be computed, taking into account that samples from frame A and frame B are drawn independently, as follows

$$\hat{V}(\hat{Y}_H) = \hat{V}(\hat{Y}_a) + \theta^2 \hat{V}(\hat{Y}_{ab}^A) + \theta \widehat{Cov}(\hat{Y}_a, \hat{Y}_{ab}^A) + (1-\theta)^2 \hat{V}(\hat{Y}_{ab}^B) + \hat{V}(\hat{Y}_b) + (1-\theta)\widehat{Cov}(\hat{Y}_b, \hat{Y}_{ab}^B), \tag{2}$$

where hats denote suitable variance and covariance estimators.

Fuller and Burmeister (1972) introduce information from the estimation of overlap domain size, obtaining the following estimator

$$\hat{Y}_{FB} = \hat{Y}_a + \hat{Y}_b + \beta_1 \hat{Y}_{ab}^A + (1-\beta_1)\hat{Y}_{ab}^B + \beta_2(\hat{N}_{ab}^A - \hat{N}_{ab}^B), \tag{3}$$

where $\hat{N}_{ab}^A = \sum_{k \in s_A} \delta_k(ab) d_k^A$ and $\hat{N}_{ab}^B = \sum_{k \in s_B} \delta_k(ab) d_k^B$. Fuller and Burmeister (1972) also show that

$$\begin{bmatrix} \tilde{\beta}_1 \\ \tilde{\beta}_2 \end{bmatrix} = - \begin{bmatrix} V(\hat{Y}_{ab}^A - \hat{Y}_{ab}^B) & Cov(\hat{Y}_{ab}^A - \hat{Y}_{ab}^B, \hat{N}_{ab}^A - \hat{N}_{ab}^B) \\ Cov(\hat{Y}_{ab}^A - \hat{Y}_{ab}^B, \hat{N}_{ab}^A - \hat{N}_{ab}^B) & V(\hat{N}_{ab}^A - \hat{N}_{ab}^B) \end{bmatrix}^{-1} \times \begin{bmatrix} Cov(\hat{Y}_a + \hat{Y}_b + \hat{Y}_{ab}^B, \hat{Y}_{ab}^A - \hat{Y}_{ab}^B) \\ Cov(\hat{Y}_a + \hat{Y}_b + \hat{Y}_{ab}^B, \hat{N}_{ab}^A - \hat{N}_{ab}^B) \end{bmatrix}$$

are the optimal values for β_1 and β_2 in the sense that they minimize the variance of the estimator. Again, $\tilde{\beta}_1$ and $\tilde{\beta}_2$ need to be estimated, since population values are not known in practice. An estimator for the variance of $\hat{Y}_{FB}$ is given by

$$\hat{V}(\hat{Y}_{FB}) = \hat{V}(\hat{Y}_a) + \hat{V}(\hat{Y}_B) + \hat{\beta}_1(\widehat{Cov}(\hat{Y}_a, \hat{Y}_{ab}^A) - \widehat{Cov}(\hat{Y}_B, \hat{Y}_{ab}^B)) + \hat{\beta}_2(\widehat{Cov}(\hat{Y}_a, \hat{N}_{ab}^A) - \widehat{Cov}(\hat{Y}_B, \hat{N}_{ab}^B)), \tag{4}$$

with $\hat{Y}_B = \hat{Y}_b + \hat{Y}_{ab}^B$.

Bankier (1986) and Kalton and Anderson (1986) combine all sampling units coming from the two frames, s_A and s_B, trying to build a single sample as if it was drawn from only one frame. Sampling weights for the units in the overlap domain need, then, to be modified to avoid bias. These adjusted weights are

$$\tilde{d}_k^A = \begin{cases} d_k^A & \text{if } k \in a \\ (1/d_k^A + 1/d_k^B)^{-1} & \text{if } k \in ab \end{cases} \quad \text{and} \quad \tilde{d}_k^B = \begin{cases} d_k^B & \text{if } k \in b \\ (1/d_k^A + 1/d_k^B)^{-1} & \text{if } k \in ab \end{cases}$$

or, summarizing,

$$\tilde{d}_k = \begin{cases} d_k^A & \text{if } k \in a \\ (1/d_k^A + 1/d_k^B)^{-1} & \text{if } k \in ab \\ d_k^B & \text{if } k \in b \end{cases}. \tag{5}$$

Hence, the estimator can be expressed in the form

$$\hat{Y}_{BKA} = \sum_{k \in s_A} \tilde{d}_k^A y_k + \sum_{k \in s_B} \tilde{d}_k^B y_k = \sum_{k \in s} \tilde{d}_k y_k, \tag{6}$$

with $s = s_A \cup s_B$. Note that to compute this estimator, one needs to know, for units in the sample coming from the overlap domain, the inclusion probability under both sampling designs. Rao and Skinner (1996) propose the following unbiased estimator for the variance of the estimator

$$\hat{V}(\hat{Y}_{BKA}) = \hat{V}(\sum_{k \in s_A} \tilde{z}_k^A) + \hat{V}(\sum_{k \in s_B} \tilde{z}_k^B), \tag{7}$$

where $\tilde{z}_k^A = \delta_k(a)y_k + (1-\delta_k(a))y_k \frac{\pi_k^A}{\pi_k^A + \pi_k^B}$ and $\tilde{z}_k^B = \delta_k(b)y_k + (1-\delta_k(b))y_k \frac{\pi_k^B}{\pi_k^A + \pi_k^B}$.

When frame sizes, N_A and N_B, are known, estimator (6) can be adjusted to increase efficiency through different procedures such as, for example, raking ratio (Bankier, 1986; Skinner, 1991). Applying the latter, one obtains a new estimator, usually called raking ratio (Skinner, 1991), which has the form

$$\hat{Y}_{SFRR} = \frac{N_A - \hat{N}_{ab}^{rake}}{\hat{N}_a}\hat{Y}_a^A + \frac{N_B - \hat{N}_{ab}^{rake}}{\hat{N}_b}\hat{Y}_b^B + \frac{\hat{N}_{ab}^{rake}}{\hat{N}_{abS}}\hat{Y}_{abS}, \tag{8}$$

where $\hat{Y}_{abS} = \sum_{k \in s_A} \tilde{d}_k^A \delta_k(ab) y_k + \sum_{k \in s_B} \tilde{d}_k^B \delta_k(ab) y_k$, $\hat{N}_{abS} = \sum_{k \in s_A} \tilde{d}_k^A \delta_k(ab) + \sum_{k \in s_B} \tilde{d}_k^B \delta_k(ab)$, $\hat{N}_a = \sum_{k \in s_A} \delta_k(a)$, $\hat{N}_b = \sum_{k \in s_B} \delta_k(b)$ and $\hat{N}_{ab}^{rake}$ is the smaller root of the quadratic equation $\hat{N}_{abS}x^2 - (\hat{N}_{abS}(N_A + N_B) + \hat{N}_{aS}^A \hat{N}_{bS}^B)x + \hat{N}_{abS}N_A N_B = 0$.

Skinner and Rao (1996) use a pseudo maximum likelihood approach to extend to complex designs the maximum likelihood estimator proposed by Fuller and Burmeister (1972) only for simple random sampling without replacement. The resulting estimator is given by

$$\hat{Y}_{PML} = \frac{N_A - \hat{N}_{ab}^{PML}(\gamma)}{\hat{N}_a^A}\hat{Y}_a^A + \frac{N_B - \hat{N}_{ab}^{PML}(\gamma)}{\hat{N}_b^B}\hat{Y}_b^B + \frac{\hat{N}_{ab}^{PML}(\gamma)}{\gamma \hat{N}_{ab}^A + (1-\gamma)\hat{N}_{ab}^B}[\gamma \hat{Y}_{ab}^A + (1-\gamma)\hat{Y}_{ab}^B], \tag{9}$$

where $\hat{N}_{ab}^{PML}(\gamma)$ is the smallest of the roots of the quadratic equation $[\gamma/N_B + (1-\gamma)/N_A]x^2 - [1 + \gamma \hat{N}_{ab}^A/N_B + (1-\gamma)\hat{N}_{ab}^B/N_A]x + \gamma \hat{N}_{ab}^A + (1-\gamma)\hat{N}_{ab}^B = 0$ and $\gamma \in (0,1)$. It is also shown that the following value for γ

$$\gamma_{opt} = \frac{\hat{N}_a N_B V(\hat{N}_{ab}^B)}{\hat{N}_a N_B V(\hat{N}_{ab}^B) + \hat{N}_b N_A V(\hat{N}_{ab}^A)} \tag{10}$$

minimizes the variance of $\hat{Y}_{PML}$. One can use the delta method to obtain a consistent estimator of the variance of this estimator of the form

$$\hat{V}(\hat{Y}_{PML}) = \hat{V}(\sum_{k \in s_A} \tilde{z}_k^A) + \hat{V}(\sum_{k \in s_B} \tilde{z}_k^B), \tag{11}$$

where, in this case, $\tilde{z}_k^A = y_k - \frac{\hat{Y}_a}{\hat{N}_a}$ if $k \in a$ and $\tilde{z}_k^A = \hat{\gamma}_{opt}\left(y_k - \frac{\hat{Y}_{ab}^A}{\hat{N}_{ab}^A}\right) + \hat{\lambda}\hat{\phi}$ if $k \in ab$, with $\hat{\gamma}_{opt}$ an estimator of γ_{opt} in (10) obtained by replacing population quantities with their estimators, $\hat{\lambda} = \frac{n_A/N_A \hat{Y}_{ab}^A + n_B/N_B \hat{Y}_{ab}^B}{n_A/N_A \hat{N}_{ab}^A + n_B/N_B \hat{N}_{ab}^B} - \frac{\hat{Y}_a}{\hat{N}_a} - \frac{\hat{Y}_b}{\hat{N}_b}$ and $\hat{\phi} = \frac{n_A \hat{N}_b}{n_A \hat{N}_b + n_B \hat{N}_a}$. Similary, one can define $\tilde{z}_k^B = y_k - \frac{\hat{Y}_b}{\hat{N}_b}$ if $k \in b$ and $\tilde{z}_k^B = (1-\hat{\gamma}_{opt})\left(y_k - \frac{\hat{Y}_{ab}^B}{\hat{N}_{ab}^B}\right) + \hat{\lambda}(1-\hat{\phi})$ if $k \in ab$.

More recently, Rao and Wu (2010) proposed a pseudo empirical likelihood estimator for the population mean based on poststratified samples. The estimator is computed as

$$\hat{\bar{Y}}_{PEL} = \frac{N_a}{N}\hat{\bar{Y}}_a + \frac{\eta N_{ab}}{N}\hat{\bar{Y}}_{ab}^A + \frac{(1-\eta)N_{ab}}{N}\hat{\bar{Y}}_{ab}^B + \frac{N_b}{N}\hat{\bar{Y}}_b, \tag{12}$$

where, in this case, $\hat{\bar{Y}}_a = \sum_{k \in s_A} \hat{p}_{ak} y_k \delta_k(a)$, $\hat{\bar{Y}}_{ab}^A = \sum_{k \in s_A} \hat{p}_{abk}^A y_k \delta_k(ab)$, $\hat{\bar{Y}}_{ab}^B = \sum_{k \in s_B} \hat{p}_{abk}^B y_k \delta_k(ab)$ and $\hat{\bar{Y}}_b = \sum_{k \in s_B} \hat{p}_{bk} y_k \delta_k(b)$ with $\hat{p}_{ak}$, $\hat{p}_{abk}^A$, $\hat{p}_{abk}^B$ and $\hat{p}_{bk}$ the weights resulting from maximizing the pseudo empirical likelihood procedure under a set of constraints (see Rao and Wu 2010 for details). Furthermore, $\eta \in (0,1)$. In this case, it is assumed that N_A, N_B and N_{ab} are known, but this is not always the case. The authors also provide modifications to be carried out in (12) to adapt it to situations where only N_A and N_B are known or where none of N_A, N_B or N_{ab} are known. In addition, auxiliary information coming from either one or both frames can be incorporated into the estimation process to improve the accuracy of the estimates. In addition, instead of an analytic form for the variance of this estimator, Rao and Wu (2010) propose to compute confidence intervals using the bi-section method described by Wu (2005) for one single frame and extending it to the dual frame case. This method constructs intervals of the form $\{\theta | r_{ns}(\theta) < \chi_1^2(\alpha)\}$, where $\chi_1^2(\alpha)$ is the $1-\alpha$ quantile from a χ^2 distribution with one degree of freedom and $r_{ns}(\theta)$ represents the so called pseudo empirical log likelihood ratio statistic, which can be obtained as a difference of two pseudo empirical likelihood functions.

Recently, Ranalli et al. (2013) extended calibration procedures to estimation from dual frame sampling assuming that some kind of auxiliary information is available. For example, assuming there are p auxiliary variables, $\mathbf{x}_k(x_{1k}, \ldots, x_{pk})$ is the value taken by such auxiliary variables on unit k. Each auxiliary variable may be available only for units in frame A, only for units in frame B or for units in the whole population. In addition, it is assumed that the vector of population totals of the auxiliary variables, $\mathbf{t_x} = \sum_{k\in\mathcal{U}} \mathbf{x}_k$ is also known. In this context, the dual frame calibration estimator can be defined as follows

$$\hat{Y}_{CALDF} = \sum_{k\in s} d_k^{CALDF} y_k, \tag{13}$$

where weights d_k^{CALDF} are such that $\min \sum_{k\in s} G(d_k^{CALDF}, \check{d}_k)$ subject to $\sum_{k\in s} d_k^{CALDF}\mathbf{x}_k = \mathbf{t_x}$, with $G(\cdot,\cdot)$ a determined distance measure and

$$\check{d}_k = \begin{cases} d_k^A & \text{if } k \in a \\ \eta d_k^A & \text{if } k \in ab \bigcap s_A \\ (1-\eta) d_k^B & \text{if } k \in ab \bigcap s_B \\ d_k^B & \text{if } k \in b \end{cases}, \tag{14}$$

where $\eta \in [0,1]$.

Then, with a similar approach to that of $\hat{Y}_{BKA}$, another calibration estimator can be computed as

$$\hat{Y}_{CALSF} = \sum_{k\in s} d_k^{CALSF} y_k, \tag{15}$$

with weights d_k^{CALSF} verifying that $\min \sum_{k\in s} G(d_k^{CALSF}, \tilde{d}_k)$ subject to $\sum_{k\in s} d_k^{CALSF}\mathbf{x}_k = \mathbf{t_x}$, being $\tilde{d}_k$ the weights defined in (5).

An estimator of the variance of any calibration estimator can be obtained using Deville's method (Deville, 1993) through the following expression

$$\hat{V}(\hat{Y}) = \frac{1}{1-\sum_{k\in s} a_k^2} \sum_{k\in s} \frac{d_k^\star - 1}{d_k^\star} \left(d_k^\star e_k - \sum_{l\in s} a_l d_l^\star e_l \right)^2, \tag{16}$$

where $d_k^\star$ is given by (5) or by (14) according to whether we use $\hat{Y}_{CALSF}$ or $\hat{Y}_{CALDF}$, respectively. In addition, $a_k = \frac{d_k^\star - 1}{d_k^\star} / \sum_{l\in s} \frac{d_l^\star - 1}{d_l^\star}$ and e_k are the residuals of the generalized regression of y on $\mathbf{x}$.

Some of the estimators described above are particular types of calibration estimators. For example, estimator (8) can be obtained as a particular case of $\hat{Y}_{CALSF}$ in the case where frame sizes N_A and N_B are known and the `"raking"` method is selected for calibration. Having noted this, one can use (16) to calculate an estimator of variance of (8). See Ranalli et al. (2013) for more details.

Table 1 shows a summary of the previous dual frame estimators according to the auxiliary information required. It can be noted that Hartley, FB and BKA estimators can be computed even when no information is available, but they cannot incorporate some auxiliary information when available. PML and SFRR can incorporate information on N_A and N_B, but PEL and CAL type estimators are the most flexible in that they can incorporate any kind of auxiliary information available.

Jackknife variance estimation

Variance estimation methods exposed so far depend on each specific estimator, so comparisons between variance estimations may lead to incorrect conclusions. Instead, one can consider jackknife, originally proposed by Quenouille (1949, 1956) (see Wolter 2007 for a detailed description of this method in survey sampling) and extended to dual frame surveys by Lohr and Rao (2000), which can be used to estimate variances irrespective of the type of estimator allowing us to compare estimated efficiency for different estimators.

For a non-stratified design in each frame, the jackknife estimator of the variance for any of the estimators described, generically denoted by $\hat{Y}_c$, is given by

$$v_J(\hat{Y}_c) = \frac{n_A - 1}{n_A} \sum_{i\in s_A} (\hat{Y}_c^A(i) - \overline{Y}_c^A)^2 + \frac{n_B - 1}{n_B} \sum_{j\in s_B} (\hat{Y}_c^B(j) - \overline{Y}_c^B)^2, \tag{17}$$

with $\hat{Y}_c^A(i)$ the value of estimator $\hat{Y}_c$ after dropping unit i from s_A and $\overline{Y}_c^A$ the mean of values $\hat{Y}_c^A(i)$. Similarly, one can define $\hat{Y}_c^B(j)$ and $\overline{Y}_c^B$.

Jackknife may present an important bias when designs are without replacement. One could,

	None	*N_A, N_B known*	*N_a, N_b and N_{ab} known*	*N_a, N_{ab}, N_b and X_A and/or X_B known*
Hartley	✓			
FB	✓			
PML		✓		
PEL	✓	✓	✓	✓
CalDF	✓	✓	✓	✓
BKA*	✓			
SFRR*		✓		
CalSF*	✓	✓	✓	✓

(*) Inclusion probabilities are known in overlap domain *ab* for both frames.

Table 1: Estimator's capabilities versus auxiliary information availability

then, incorporate an approximate finite-population correction to estimation to achieve unbiasedness. For example, assuming that a finite-population correction is needed in frame A, a modified jackknife estimator of variance, $v_J^*(\hat{Y}_c)$, can be calculated by replacing $\hat{Y}_c^A(i)$ in (17) with $\hat{Y}_c^{A*}(i) = \hat{Y}_c + \sqrt{1-\overline{\pi}_A}(\hat{Y}_c^A(i) - \hat{Y}_c)$, where $\overline{\pi}_A = \sum_{k \in s_A} \pi_k^A / n_A$.

Consider now a stratified design in each frame, where frame A is divided into H strata and frame B is divided into L strata. From stratum h of frame A, a sample of n_{Ah} units from the N_{Ah} population units in the stratum is drawn. Similarly, in stratum l of frame B, one selects n_{Bl} units from the N_{Bl} composing the stratum. The jackknife estimator of the variance can be defined, then, as follows

$$v_J(\hat{Y}_c) = \sum_{h=1}^{H} \frac{n_{Ah}-1}{n_{Ah}} \sum_{i \in s_{Ah}} (\hat{Y}_c^A(hi) - \overline{Y}_c^{Ah})^2 + \sum_{l=1}^{L} \frac{n_{Bl}-1}{n_{Bl}} \sum_{i \in s_{Bl}} (\hat{Y}_c^B(lj) - \overline{Y}_c^{Bl})^2, \tag{18}$$

where $\hat{Y}_c^A(hi)$ is the value taken by $\hat{Y}_c$ after dropping unit i of stratum h from sample s_{Ah} and $\overline{Y}_c^{Ah}$ is the mean of values $\hat{Y}_c^A(hi)$. $\hat{Y}_c^B(lj)$ and $\overline{Y}_c^{Bl}$ can be defined in a similar way. Again, one can include an approximate finite-population correction in any stratum needing it. In case of a non stratified design in one frame and a stratified design in the other one, previous methods can be combined to obtain the corresponding jackknife estimator of the variance.

Stratified cluster sampling is very common in practice. Now we illustrate the jackknife estimator when a stratified sample of clusters is selected. Suppose that frame A has H strata and stratum h has N_{Ah} observation units and $\widetilde{N}_{Ah}$ primary sampling units (clusters), of which $\widetilde{n}_{Ah}$ are sampled. Frame B has L strata, and stratum l has N_{Bl} observation units and $\widetilde{N}_{Bh}$ primary sampling units, of which $\widetilde{n}_{Bl}$ are sampled.

To define the jackknife estimator of the variance, let $\widetilde{Y}_c^A(hj)$ be the estimator of the same form as $\hat{Y}_c$ when the observations of sample primary sampling unit j of stratum h from sample in frame A are omitted. Similarly, $\widetilde{Y}_c^B(lk)$ is of the same form as $\hat{Y}_c$ when the observations of sample primary sampling unit k of stratum l from sample in frame B are omitted. The jackknife variance estimator is then given by:

$$v_J(\hat{Y}_c) = \sum_{h=1}^{H} \frac{\widetilde{n}Ah-1}{\widetilde{n}_{Ah}} \sum_{j=1}^{\widetilde{n}_{Ah}} (\widetilde{Y}_c^A(hj) - \widetilde{Y}_c^{Ah})^2 + \sum_{l=1}^{L} \frac{\widetilde{n}_{Bl}-1}{\widetilde{n}_{Bl}} \sum_{k \in s_{Bl}} (\widetilde{Y}_c^B(lk) - \widetilde{Y}_c^{Bl})^2, \tag{19}$$

where $\widetilde{Y}_c^{Ah}$ is the mean of values $\widetilde{Y}_c^A(hj)$ and $\widetilde{Y}_c^{Bl}$ is the mean of values $\widetilde{Y}_c^B(lk)$.

The R package Frames2

Frames2 is a new R package for point and interval estimation from dual frame sampling. It consists of eight main functions (`Hartley`, `FB`, `BKA`, `SFRR`, `PML`, `PEL`, `CalSF` and `CalDF`), each of them implementing one of the estimators described in the previous sections. The package also includes an additional function called `Compare` which provides a summary with all possible estimators that can be computed from the information provided as input. Moreover, six extra functions implementing auxiliary operations, like computation of Horvitz-Thompson estimators or of the covariance between two

	user	system	elapsed
	$n_A = 10605, n_B = 13635$		
Hartley	0.01	0.02	0.04
FB	0.05	<0.01	0.07
BKA	0.03	<0.01	0.05
PML	0.02	0.02	0.03
SFRR	0.03	0.03	0.07
CalSF	0.03	<0.01	0.06
CalDF	0.04	0.01	0.05
	$n_A = 105105, n_B = 135135$		
Hartley	0.11	0.06	0.19
FB	0.27	0.07	0.32
BKA	0.13	0.05	0.17
PML	0.16	0.02	0.18
SFRR	0.42	0.12	0.54
CalSF	0.20	0.08	0.30
CalDF	0.22	0.07	0.31

Table 2: User, system and elapsed times (in seconds) for estimators considering different sample sizes.

Horvitz-Thompson estimators, have also been included in the package to achieve a more understandable code. Finally, the package includes eight more functions, one for each estimator, for the calculation of confidence intervals based on the jackknife variance estimator.

A notable feature of these functions is the strong argument check. Functions check general aspects such as the presence of `NA` or `NaN` values in its arguments, the number of main variables considered in the frames (that should match), the length of the arguments in each frame (that should also match) and the values for arguments indicating the domain each unit belongs to (which only can be `"a"` or `"ab"` for frame A or `"b"` or `"ba"` for frame B). If any issue is encountered, the function displays an error message indicating what the problem is and what argument causes it, so that the user can manage errors easily. Furthermore, each function has additional checks depending on its specific characteristics or arguments. The main aim of this exhaustive check is to guarantee validity of the arguments, so one can avoid, to the extent possible, issues during computation.

Much attention has also been devoted to computational efficiency. Frequently, populations in a survey are extremely large or it is needed to keep sampling error below a certain value. As a consequence, one needs to consider large sample sizes, often in the order of tens of thousands sampling units. In these situations, computational efficiency of functions is essential, particularly when several variables are considered. Otherwise, the user can face high runtimes and heavy computational loads. In this sense, functions of **Frames2** are developed according to strict efficiency measures, using the power of R to use matrix calculation to avoid loops and increase the computational efficiency.

Table 2 shows user and system times necessary to compute estimators using an Intel(R) Core(TM) i7-3770 at 3.40 GHz when different sample sizes are considered. Elapsed time is also included to get an idea about the real time user needs to get estimations.

Functions of **Frames2** have been implemented from an user-oriented perspective to increase usability. In this sense, most input parameters (which are the communication channel between the user and the function) are divided into two groups, depending on the frame they come from. This is to adapt functions as much as possible to the usual estimation procedure, in which the first step is to draw two independent samples, one from each frame. On the other hand, estimation details are managed internally by functions so that they are not visible for the user, who does not need to manage them.

Construction of functions has been carried out so that they perform properly in as many situations as possible. As noted in the introductory section, one can face several situations when using two sampling frames depending on their relative positions. Although the most common is the one depicted in Figure 1, cases shown in Figures 2 and 3 may arise as well. All estimators described except PEL can be modified to cover these three situations, so corresponding functions of **Frames2** include necessary changes to produce estimates irrespective of the situation.

On the other hand, it is usual, when conducting a survey, to collect information on many variables of interest. To adapt to such situations, all functions are programmed to produce estimates when there are more than one variable of interest with only one call. To this end, parameters containing information about main variables observed in each frame can be either vectors, when only one variable is considered or matrices or data frames, when there are several variables under study. Cases in which the main aim of the survey is the estimation of population means or proportions are also very frequent. Hence, from the estimation of the population total for a variable, functions compute estimation of the mean as $\hat{\bar{Y}} = \hat{Y}/\hat{N}$. To obtain the estimation of the population size, functions internally apply the estimation procedure at issue to indicator vectors $\mathbf{1}_A$ and $\mathbf{1}_B$ of sizes n_A and n_B, respectively.

To get maximum flexibility, functions have been programmed to calculate estimates in cases in which the user disposes of first and second order inclusion probabilities and in those other in which only first order ones are available, indistinctly. Knowledge of both first and second order inclusion probabilities is a strong assumption that does not always occur in practice. However, when calculating most of the estimators described in previous sections, second order inclusion probabilities are needed in many steps of the estimation procedure, mainly in computing estimated variances of a Horvitz-Thompson estimator or estimated covariances between two Horvitz-Thompson estimators. As an alternative, one can obtain variance estimations from only first order inclusion probabilities applying Deville's method reported in (16), by substituting residuals e_k with the values of the variable of interest, y_k. Covariance estimations are also obtained from variances through the following expression

$$\widehat{Cov}(\hat{Y}, \hat{X}) = \frac{\hat{V}(\hat{Y} + \hat{X}) - \hat{V}(\hat{Y}) - \hat{V}(\hat{X})}{2}.$$

To cover both cases, the user has the possibility to consider different data structures for parameters relating to inclusion probabilities. So, if both first and second order inclusion probabilities are available, these parameters will be square matrices, whereas if only first order inclusion probabilities are known, these arguments will be vectors. The only restriction here is that the type of both should match.

As can be deduced from previous sections, an essential aspect when computing estimates in dual frames is to know the domain each unit belongs to. Character vectors `domains_A` and `domains_B` are used for this purpose. The former can take values `"a"` or `"ab"`, while the latter can take values `"b"` or `"ba"`. Any other value will be considered as incorrect.

Data description

To illustrate how functions operate, we use the data sets `DatA` and `DatB`, both included in the package. `DatA` contains information about $n_A = 105$ households selected through a stratified sampling design from the $N_A = 1735$ households forming frame A. More specifically, frame A has been divided into 6 strata of sizes $N_{hA} = (727, 375, 113, 186, 115, 219)$ from which simple random without replacement samples of sizes $n_{hA} = (15, 20, 15, 20, 15, 20)$ have been drawn. On the other hand, a simple random without replacement sample of $n_B = 135$ households has been selected from the $N_B = 1191$ households in frame B. The size of the overlap domain for this case is $N_{ab} = 601$. This situation is depicted in Figure 4.

Both data sets contain information about the same variables. To better understand their structure, we report the first three rows of `DatA`:

```
> library(Frames2)
> data(DatA)
> head(DatA, 3)
  Domain   Feed   Clo   Lei     Inc    Tax     M2 Size      ProbA     ProbB Stratum
1      a 194.48 38.79 23.66 2452.07 112.90   0.00    0 0.02063274 0.0000000       1
2      a 250.23 16.92 22.68 2052.37 106.99   0.00    0 0.02063274 0.0000000       1
3     ab 199.95 24.50 23.24 2138.24 121.16 127.41    2 0.02063274 0.1133501       1
```

Each data set incorporates information about three main variables: Feeding, Clothing and Leisure. Additionally, there are two auxiliary variables for the units in frame A (Income and Taxes) and another two variables for units in frame B (Metres2 and Size). Corresponding totals for these auxiliary variables are assumed known in the entire frame and they are $T^A_{Inc} = 4300260, T^A_{Tax} = 215577, T^B_{M2} = 176553$ and $T^B_{Size} = 3529$. Finally, a variable indicating the domain each unit belongs to and two variables showing the first order inclusion probabilities for each frame complete the data sets.

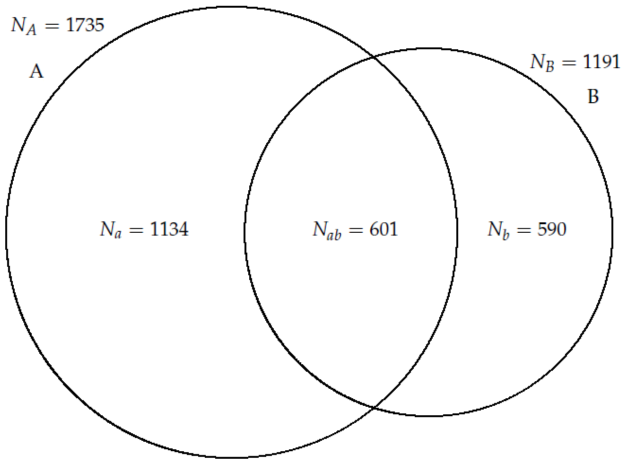


Figure 4: Frame and domain sizes for the data sets.

Numerical square matrices PiklA and PiklB, with dimensions $n_A = 105$ and $n_B = 135$, are also used as probability inclusion matrices. These matrices contains second order inclusion probabilities and first order inclusion probabilities as diagonal elements. To check the appearance of these matrices the first submatrix of order 6 of PiklA is shown.

```
> data(PiklA)
> PiklA[1:6, 1:6]
            [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
[1,] 0.020632737 0.000397876 0.000397876 0.000397876 0.000397876 0.000397876
[2,] 0.000397876 0.020632737 0.000397876 0.000397876 0.000397876 0.000397876
[3,] 0.000397876 0.000397876 0.020632737 0.000397876 0.000397876 0.000397876
[4,] 0.000397876 0.000397876 0.000397876 0.020632737 0.000397876 0.000397876
[5,] 0.000397876 0.000397876 0.000397876 0.000397876 0.020632737 0.000397876
[6,] 0.000397876 0.000397876 0.000397876 0.000397876 0.000397876 0.020632737
```

No auxiliary information

When there is no further information than the one on the variables of interest, one can calculate some of the estimators described in previous sections (as, for example, (1) or (3)) as follows

```
> library(Frames2)
>
> data(DatA)
> data(DatB)
> data(PiklA)
> data(PiklB)
>
> yA <- with(DatA, data.frame(Feed, Clo))
> yB <- with(DatB, data.frame(Feed, Clo))
>
> ## Estimation for variables Feeding and Clothing using Hartley and
> ## Fuller-Burmeister estimators with first and second order probabilities known
> Hartley(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain)

Estimation:
             Feed         Clo
Total 586959.9820 71967.62214
Mean     246.0429    30.16751
> FB(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain)

Estimation:
             Feed         Clo
```

```
Total 591665.5078 72064.99223
Mean     248.0153    30.20832
> ## This is how estimates change when only first order probabilities are considered
> Hartley(yA, yB, DatA$ProbA, DatB$ProbB, DatA$Domain, DatB$Domain)

Estimation:
             Feed         Clo
Total 570867.8042 69473.86532
Mean     247.9484    30.17499
> FB(yA, yB, DatA$ProbA, DatB$ProbB, DatA$Domain, DatB$Domain)

Estimation:
             Feed         Clo
Total 571971.9511 69500.11448
Mean     248.4279    30.18639
```

As the result, an object of class "EstimatorDF" is returned, showing, by default, estimations for the population total and mean for the two considered variables. In general, m columns will be displayed, one for each of the m variables estimated. Further information about the estimation process (as variance estimations or values of parameters involved in estimation) can be displayed by using function summary.

```
> summary(Hartley(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain))

Call:
Hartley(ysA = yA, ysB = yB, pi_A = PiklA, pi_B = PiklB, domains_A = DatA$Domain,
    domains_B = DatB$Domain)

Estimation:
             Feed         Clo
Total 586959.9820 71967.62214
Mean     246.0429    30.16751

Variance Estimation:
                   Feed          Clo
Var. Total 2.437952e+08 4.728875e+06
Var. Mean  4.283804e+01 8.309261e-01

Total Domain Estimations:
                  Feed      Clo
Total dom. a  263233.1 31476.84
Total dom. ab 166651.7 21494.96
Total dom. b  164559.2 20451.85
Total dom. ba 128704.7 15547.49

Mean Domain Estimations:
                 Feed      Clo
Mean dom. a  251.8133 30.11129
Mean dom. ab 241.6468 31.16792
Mean dom. b  242.2443 30.10675
Mean dom. ba 251.5291 30.38466

Parameters:
           Feed       Clo
theta 0.8027766 0.7551851
```

The previous output shows in the component Estimation the estimations of the population total and the population mean computed using the Harley estimator, that is, $\hat{Y}_H$ and $\hat{\bar{Y}}_H$. Estimated variances of these estimations, $\hat{V}(\hat{Y}_H)$ and $\hat{V}(\hat{\bar{Y}}_H)$, are displayed in component Variance Estimation. In the section Total Domain Estimations we can see estimations $\hat{Y}_a$, $\hat{Y}_{ab}^A$, $\hat{Y}_b$ and $\hat{Y}_{ab}^B$. Estimates for the population mean for each domain, $\hat{\bar{Y}}_a$, $\hat{\bar{Y}}_{ab}^A$, $\hat{\bar{Y}}_b$ and $\hat{\bar{Y}}_{ab}^B$ are displayed in the component Mean Domain Estimations. Finally, $\hat{\theta}$, the estimated value of the parameter involved in computation of the Hartley estimator is shown.

This additional information depends on the way each estimator is formulated. Thus, for example, extra information will include a parameter component when applied to a call to the Fuller-Burmeister estimator (and values of estimates for β_1 and β_2 will be displayed there), but not when applied to a call to the Bankier-Kalton-Anderson estimator (because no parameters are used when computing this estimator).

Results slightly change when a confidence interval is required. In that case, the user has to indicate the confidence level desired for the interval through argument `conf_level` (default is `NULL`) and add it to the list of input parameters. The function calculates, then, a confidence interval based on the pivotal method. This method yields a confidence interval as follows: $\hat{Y} \pm z_{\alpha/2}\sqrt{\hat{V}(\hat{Y})}$ where $z_{\alpha/2}$ is the critical value of a standard normal distribution. Only for the case of PEL, confidence intervals are based on a χ^2 distribution and the bi-section method (Rao and Wu, 2010). In this case, default output will show 6 rows for each variable, lower and upper boundaries for confidence intervals are displayed together with estimates. So, one can obtain a 95% confidence interval for estimations in the last two of the previous four cases in this way.

```
> Hartley(yA, yB, DatA$ProbA, DatB$ProbB, DatA$Domain, DatB$Domain, 0.95)

Estimation and  95 % Confidence Intervals:
                   Feed         Clo
Total       570867.8042 69473.86532
Lower Bound 511904.6588 61756.37677
Upper Bound 629830.9496 77191.35387
Mean           247.9484    30.17499
Lower Bound    222.3386    26.82301
Upper Bound    273.5582    33.52697
> FB(yA, yB, DatA$ProbA, DatB$ProbB, DatA$Domain, DatB$Domain, 0.95)

Estimation and  95 % Confidence Intervals:
                   Feed         Clo
Total       571971.9511 69500.11448
Lower Bound 513045.7170 61802.57411
Upper Bound 630898.1852 77197.65484
Mean           248.4279    30.18639
Lower Bound    222.8342    26.84307
Upper Bound    274.0217    33.52971
```

For estimators constructed as (6), numeric vectors `pik_ab_B` and `pik_ba_A` of lengths n_A and n_B should be added as arguments. While `pik_ab_B` represents first order inclusion probabilities according to sampling design in frame B for units belonging to the overlap domain selected in the sample drawn from frame A, `pik_ba_A` contains first order inclusion probabilities according to the sampling design in frame A for units belonging to the overlap domain selected in sample drawn from frame B.

```
> yA <- with(DatA, data.frame(Feed, Clo, Lei))
> yB <- with(DatB, data.frame(Feed, Clo, Lei))
>
> ## Bankier-Kalton-Anderson estimation and a 95% confidence
> ## interval for the three main variables
> BKA(yA, yB, DatA$ProbA, DatB$ProbB, DatA$ProbB, DatB$ProbA, DatA$Domain,
+   DatB$Domain, 0.95)

Estimation and  95 % Confidence Intervals:
                   Feed         Clo         Lei
Total       566434.3200 68959.26705 50953.07583
Lower Bound 624569.2139 76538.11015 56036.23578
Upper Bound 508299.4262 61380.42395 45869.91588
Mean           247.8845    30.17814    22.29822
Lower Bound    273.3257    33.49482    24.52273
Upper Bound    222.4434    26.86147    20.07372
```

Note that these examples include just a few of the estimators that can be used when no auxiliary information is known. As noted in Table 1, other estimators, as those in (12), (13) or (15), can be also calculated in this case. In this context, function `Compare` is quite useful, since it returns all possible estimators that can be computed according to the information provided as input.

```
> Compare(yA, yB, DatA$ProbA, DatB$ProbB, DatA$Domain, DatB$Domain)
$Hartley
```

```
Estimation:
             Feed         Clo        Lei
Total 570867.8042 69473.86532 51284.2727
Mean     247.9484    30.17499    22.2746

$FullerBurmeister

Estimation:
             Feed         Clo         Lei
Total 571971.9511 69500.11448 51210.03819
Mean     248.4279    30.18639    22.24236

$PEL

Estimation:
              Feed          Clo          Lei
Total 1.791588e+08 2.663164e+06 1.455533e+06
Mean  2.479314e+02 3.011373e+01 2.235969e+01

$Calibration_DF

Estimation:
             Feed         Clo        Lei
Total 595162.2604 72214.13351 53108.5059
Mean     248.8422    30.19332    22.2051
```

Using appropriate indicator variables as variables of interest, one can also estimate the overlap domain size, as shown below:

```
> indA <- as.integer(DatA$Domain == "ab")
> indB <- as.integer(DatB$Domain == "ba")
>
> Hartley(indA, indB, DatA$ProbA, DatB$ProbB, DatA$Domain, DatB$Domain)

Estimation:
             [,1]
Total 534.2743208
Mean    0.2320545
> BKA(indA, indB, DatA$ProbA, DatB$ProbB, DatA$ProbB, DatB$ProbA, DatA$Domain,
+   DatB$Domain)

Estimation:
             [,1]
Total 560.4121771
Mean    0.2452491
```

Auxiliary information about frame sizes

For estimators requiring frame sizes known, as (8) or (9), it is needed to incorporate two additional input arguments, `N_A` and `N_B`. There is also a group of estimators, including (12) and (15), that, even though able to provide estimations without the need of auxiliary information, can use frame sizes to improve their precision. The following examples show the performance of these estimators.

```
> ## SFRR estimator and CalSF estimator with frame sizes as auxiliary information
> ## using method "raking" for the calibration for the three main variables
> SFRR (yA, yB, DatA$ProbA, DatB$ProbB, DatA$ProbB, DatB$ProbA, DatA$Domain,
+   DatB$Domain, N_A = 1735, N_B = 1191)

Estimation:
             Feed         Clo         Lei
Total 584713.4070 71086.18669 52423.74035
Mean     248.2219    30.17743    22.25487
> CalSF(yA, yB, DatA$ProbA, DatB$ProbB, DatA$ProbB, DatB$ProbA, DatA$Domain,
+   DatB$Domain, N_A = 1735, N_B = 1191, met = "raking")

Estimation:
             Feed         Clo         Lei
Total 584713.4070 71086.18669 52423.74035
Mean     248.2219    30.17743    22.25487
```

As highlighted previously, both results match. Note that the argument `met` of the SF calibration estimator indicates the method used in the calibration procedure. The possibility of choosing the calibration method is given by the fact that computation of both SF and DF calibration estimators is based on the function `calib` from package **sampling** (Tillé and Matei, 2012), which can manage three different calibration methods, each one associated with one particular distance measure. These methods are: linear, raking and logit.

The requirement of knowing probabilities of inclusion in both frames for the units in the overlap domain may be restrictive in some cases. As an alternative, in cases where frame sizes are known but this condition is not met, it is possible to calculate dual frame estimators as (9), (12) or (13). Next, it is illustrated how to obtain some of these estimators with **Frames2**.

```
> ## Estimates for the three main variables using PML, PEL and CalDF
> ## with frame sizes as auxiliary information in PEL and CalDF
> PML(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191)

Estimation:
             Feed         Clo         Lei
Total 593085.4467 72272.73759 53287.68044
Mean     248.0966    30.23277    22.29104
> PEL(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191)

Estimation:
             Feed         Clo         Lei
Total 590425.4843 72211.61334 53258.38286
Mean     247.4958    30.26982    22.32497
> CalDF(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191)

Estimation:
             Feed         Clo         Lei
Total 587502.4374 71368.45308 52490.98852
Mean     248.7193    30.21385    22.22207
```

To calculate the `PEL` estimator, computational algorithms for the pseudo empirical likelihood method for the analysis of complex survey data presented by Wu (2005) are used.

Auxiliary information about domain sizes

In addition to the frame sizes, in some cases, it is possible to know the size of the overlap domain, N_{ab}. Generally, this considerably improves the precision of the estimates. This situation has been taken into account when constructing functions implementing estimators (12), (13) and (15), so the user can incorporate this information through parameter `N_ab`, as shown below

```
> ## Estimates for the three main variables using PEL estimator
> ## with frame sizes and overlap domain size as auxiliary information
> PEL(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191,
+   N_ab = 601)

Estimation:
             Feed         Clo         Lei
Total 575289.2186 70429.95642 51894.32490
Mean     247.4362    30.29245    22.32014
> ## Calibration estimators with the same auxiliary information
> ## Estimates do not change when raking method is used for the calibration
> CalSF(yA, yB, PiklA, PiklB, DatA$ProbB, DatB$ProbA, DatA$Domain, DatB$Domain,
+   N_A = 1735, N_B = 1191, N_ab = 601)

Estimation:
```

```
             Feed         Clo         Lei
Total 577163.6066 70173.20412 51726.19862
Mean     248.2424    30.18202    22.24783
> CalSF(yA, yB, PiklA, PiklB, DatA$ProbB, DatB$ProbA, DatA$Domain, DatB$Domain,
+   N_A = 1735, N_B = 1191, N_ab = 601, met = "raking")

Estimation:
             Feed         Clo         Lei
Total 577163.6067 70173.20414 51726.19863
Mean     248.2424    30.18202    22.24783
> CalDF(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191,
+   N_ab = 601)

Estimation:
             Feed         Clo         Lei
Total 578691.1756 70246.32319 51600.78973
Mean     248.8994    30.21347    22.19389
> CalDF(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191,
+   N_ab = 601, met = "raking")

Estimation:
             Feed         Clo         Lei
Total 578691.1763 70246.32328 51600.78979
Mean     248.8994    30.21347    22.19389
```

Note that in this case, calibration estimators provide the same results irrespective of the distance function employed. This is an interesting property that calibration estimators show only in the case in which all the domain sizes are known and used for calibration (see Deville 1993).

Auxiliary information about additional variables

On the other hand, some of the estimators are defined such that they can incorporate auxiliary information into the estimation process. This is the case for estimators (12), (13) and (15). Functions implementing them are also able to manage auxiliary information. To achieve maximum flexibility, functions implementing estimators (12), (13) and (15) are prepared to deal with auxiliary information when it is available only in frame A, only in frame B or in both frames. For instance, auxiliary information collected from frame A should be incorporated into functions through three arguments: `xsAFrameA` and `xsBFrameA`, numeric vectors, matrices or data frames (depending on the number of auxiliary variables in the frame); and `XA`, a numeric value or vector of length indicating population totals for the auxiliary variables considered in frame A. Similarly, auxiliary information in frame B is incorporated into each function through arguments `xsAFrameB`, `xsBFrameB` and `XB`. If auxiliary information is available in the whole population, this must be indicated through parameters `xsT` and `X`. In the following example, one can see how to calculate estimators using different type of auxiliary information

```
> ## PEL, CalSF and CalDF estimators for the three main variables
> ## using Income as auxiliary variable in frame A and Metres2 as auxiliary
> ## variable in frame B assuming frame sizes known
> PEL(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191,
+   xsAFrameA = DatA$Inc, xsBFrameA = DatB$Inc, xsAFrameB = DatA$M2,
+   xsBFrameB = DatB$M2, XA = 4300260, XB = 176553)

Estimation:
             Feed         Clo         Lei
Total 587742.7193 71809.56826 53094.20112
Mean     246.3713    30.10129    22.25614
> CalSF(yA, yB, PiklA, PiklB, DatA$ProbB, DatB$ProbA, DatA$Domain, DatB$Domain,
+   N_A = 1735, N_B = 1191, xsAFrameA = DatA$Inc, xsBFrameA = DatB$Inc,
+   xsAFrameB = DatA$M2, xsBFrameB = DatB$M2, XA = 4300260, XB = 176553)

Estimation:
             Feed         Clo         Lei
Total 582398.3181 70897.88438 52252.24741
Mean     247.5819    30.13922    22.21282
```

```
> CalDF(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191,
+   xsAFrameA = DatA$Inc, xsBFrameA = DatB$Inc, xsAFrameB = DatA$M2,
+   xsBFrameB = DatB$M2, XA = 4300260, XB = 176553)

Estimation:
             Feed         Clo         Lei
Total 585185.4497 71194.61148 52346.43878
Mean     247.8075    30.14866    22.16705
> ## Now, assume that overlap domain size is also known
> PEL(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191,
+   N_ab = 601, xsAFrameA = DatA$Inc, xsBFrameA = DatB$Inc,
+   xsAFrameB = DatA$M2, xsBFrameB = DatB$M2, XA = 4300260, XB = 176553)

Estimation:
             Feed         Clo         Lei
Total 572611.6997 69991.74803 51737.56089
Mean     246.2846    30.10398    22.25271
> CalSF(yA, yB, PiklA, PiklB, DatA$ProbB, DatB$ProbA, DatA$Domain, DatB$Domain,
+   N_A = 1735, N_B = 1191, N_ab = 601, xsAFrameA = DatA$Inc, xsBFrameA = DatB$Inc,
+   xsAFrameB = DatA$M2, xsBFrameB = DatB$M2, XA = 4300260, XB = 176553)

Estimation:
             Feed         Clo         Lei
Total 575636.7876 70076.78485 51628.27583
Mean     247.5857    30.14055    22.20571
> CalDF(yA, yB, PiklA, PiklB, DatA$Domain, DatB$Domain, N_A = 1735, N_B = 1191,
+   N_ab = 601, xsAFrameA = DatA$Inc, xsBFrameA = DatB$Inc,
+   xsAFrameB = DatA$M2, xsBFrameB = DatB$M2, XA = 4300260, XB = 176553)

Estimation:
             Feed        Clo         Lei
Total 576630.7609 70102.0037 51477.16737
Mean     248.0132    30.1514    22.14072
```

Interval estimation based on jackknife variance estimation

Finally, eight additional functions have been included, each of them calculating confidence intervals based on jackknife variance estimation for each estimator. To carry out variance estimation using the jackknife method, in addition to parameters to calculate each specific estimator, the user has to indicate through arguments `sdA` and `sdB` the sampling design applied in each frame. Possible values are `"srs"` (simple random sampling without replacement), `"str"` (stratified sampling), `"pps"` (probabilities proportional to size sampling), `"clu"` (cluster sampling) or `"strclu"` (stratified cluster sampling). Default is `"srs"` for both frames. If a stratified or a cluster sampling has been carried out in one of the frames, it is needed to include information about the strata or the clusters. Furthermore, the user is able to include a finite population correction factor in each frame by setting to `TRUE` the parameters `fcpA` and `fcpB`, set by default to `FALSE`. As the main purpose of the functions is to obtain confidence intervals, parameter `conf_level` is now mandatory. As noted, these functions can be used, for example, to make comparisons between efficiency of estimators, as shown in the next example.

```
> ## Confidence intervals through jackknife for the three main variables
> ## for estimators defined under the so called single frame approach with
> ## a stratified random sampling in frame A and a simple random sampling
> ## without replacement in frame B. Finite population correction factor
> ## is required for frame A
> JackBKA (yA, yB, DatA$ProbA, DatB$ProbB, DatA$ProbB, DatB$ProbA, DatA$Domain,
+   DatB$Domain, conf_level = 0.95, sdA = "str", strA = DatA$Stratum, fcpA = TRUE)
                      Feed         Clo         Lei
Total          566434.3200 68959.26705 50953.07583
Jack Upper End 610992.1346 74715.89841 54717.32664
Jack Lower End 521876.5055 63202.63570 47188.82502
Mean              247.8845    30.17814    22.29822
Jack Upper End    267.3840    32.69738    23.94555
Jack Lower End    228.3850    27.65891    20.65090
> JackSFRR(yA, yB, DatA$ProbA, DatB$ProbB, DatA$ProbB, DatB$ProbA, DatA$Domain,
+   DatB$Domain, N_A = 1735, N_B = 1191, conf_level = 0.95, sdA = "str",
+   strA = DatA$Stratum, fcpA = TRUE)
                      Feed         Clo         Lei
Total          584713.4070 71086.18669 52423.74035
Jack Upper End 619959.0338 76576.74587 55204.67760
Jack Lower End 549467.7802 65595.62751 49642.80309
Mean              248.2219    30.17743    22.25487
Jack Upper End    263.1843    32.50828    23.43543
Jack Lower End    233.2595    27.84659    21.07431
> JackCalSF(yA, yB, DatA$ProbA, DatB$ProbB, DatA$ProbB, DatB$ProbA, DatA$Domain,
+   DatB$Domain, N_A = 1735, N_B = 1191, N_ab = 601, conf_level = 0.95, sdA = "str",
+   strA = DatA$Stratum, fcpA = TRUE)
                      Feed         Clo         Lei
Total          577163.6066 70173.20412 51726.19862
Jack Upper End 599105.4275 73516.53187 53165.97439
Jack Lower End 555221.7858 66829.87636 50286.42285
Mean              248.2424    30.18202    22.24783
Jack Upper End    257.6798    31.62001    22.86709
Jack Lower End    238.8051    28.74403    21.62857
> ## Same for a selection of dual frame estimators
> JackHartley (yA, yB, DatA$ProbA, DatB$ProbB, DatA$Domain, DatB$Domain,
+   conf_level = 0.95, sdA = "str", strA = DatA$Stratum, fcpA = TRUE)
                      Feed         Clo         Lei
Total          570867.8042 69473.86532 51284.27265
Jack Upper End 610664.7131 74907.33129 54782.33083
Jack Lower End 531070.8954 64040.39934 47786.21447
Mean              247.9484    30.17499    22.27460
Jack Upper End    265.2336    32.53494    23.79393
Jack Lower End    230.6631    27.81504    20.75527
> JackPML(yA, yB, DatA$ProbA, DatB$ProbB, DatA$Domain, DatB$Domain,
+   N_A = 1735, N_B = 1191, conf_level = 0.95, sdA = "str", strA = DatA$Stratum,
+   fcpA = TRUE)
                      Feed         Clo         Lei
Total          594400.6320 72430.05834 53408.30337
Jack Upper End 626443.7529 76885.06491 56003.77592
Jack Lower End 562357.5111 67975.05176 50812.83082
Mean              248.0934    30.23115    22.29178
Jack Upper End    261.4677    32.09060    23.37509
Jack Lower End    234.7191    28.37171    21.20847
> JackCalDF(yA, yB, DatA$ProbA, DatB$ProbB, DatA$Domain, DatB$Domain, N_A = 1735,
+   N_B = 1191, N_ab = 601, conf_level = 0.95, sdA = "str", strA = DatA$Stratum,
+   fcpA = TRUE)
                      Feed         Clo         Lei
Total          578895.6961 70230.11306 51570.55683
Jack Upper End 601626.7000 73614.66702 53037.42260
Jack Lower End 556164.6921 66845.55910 50103.69107
Mean              248.9874    30.20650    22.18088
Jack Upper End    258.7642    31.66222    22.81179
Jack Lower End    239.2106    28.75078    21.54997
```

An application to a real telephone survey

In the example above data are separated into two data sets `DatA` and `DatB` containing domain information. But in practice, it is common to have a joint data set including units from both samples in which there is not a specific variable indicating the domain where each individual is placed. However, we can easily split the dataset and format it, so functions of **Frames2** can be applied. To illustrate how to do this, we are going to use dataset `Dat`, which includes some of the variables collected in a real dual frame survey.

Data included in `Dat` comes from an opinion survey on the Andalusian population with respect to immigration. This survey is conducted using telephone interviews on adults using two sampling frames: one for landlines and another one for cell phones. From the landline frame, a stratified sample of size 1919 was drawn, while from the cell phone frame, a sample of size 483 is drawn using simple random sampling without replacement. First-order inclusion probabilities were computed from a stratified random design in the landline frame and modified taking into account the number of fixed lines and adults in the household. In the cell phone frame first-order inclusion probabilities were computed and modified, given the number of cell phone numbers per individual. At the time of data collection, frame sizes of land and cell phones were 4,982,920 and 5,707,655, respectively, and the total population size was 6,350,916.

The data set includes information about 7 variables: `Drawnby`, which takes value 1 if the unit comes from the landline sample and value 2 if it comes from the cell phone sample; `Stratum`, which indicates the stratum each unit belongs to (for individuals in the cell phone frame, value of this variable is `NA`); `Opinion` the response to the question: “Do you think that immigrants currently living in Andalusia are quite a lot?” with value 1 representing “yes” and value 0 representing “no”; `Landline` and `Cell`, which record whether the unit possess a landline or a cell phone, respectively. First order inclusion probabilities are also included in the data set.

```
> data(Dat)
> head(Dat, 3)
  Drawnby Stratum Opinion Landline Cell ProbLandline ProbCell
1       1       2       0        1    1  0.000673623 8.49e-05
2       1       5       1        1    1  0.002193297 5.86e-05
3       1       1       0        1    1  0.001831489 7.81e-05
```

From the data of this survey we wish to estimate the number of people in Andalusia thinking that immigrants currently living in this region are quite a lot. In order to use functions of package **Frames2**, we need to split this dataset. The variables we will use to do this are `Drawnby` and `Landline` and `Cell`.

```
> attach(Dat)
> ## We can split the original dataset in four new different
> ## datasets, each one corresponding to one domain.
>
> DomainOnlyLandline <- Dat[Landline == 1 & Cell == 0,]
> DomainBothLandline <- Dat[Drawnby == 1 & Landline == 1 & Cell == 1,]
> DomainOnlyCell <- Dat[Landline == 0 & Cell == 1,]
> DomainBothCell <- Dat[Drawnby == 2 & Landline == 1 & Cell == 1,]
>
> ## From the domain datasets, we can build frame datasets
>
> FrameLandline <- rbind(DomainOnlyLandline, DomainBothLandline)
> FrameCell <- rbind(DomainOnlyCell, DomainBothCell)
>
> ## Finally, we only need to label domain of each unit using "a", "b",
> ## "ab" or "ba"
>
> Domain <- c(rep("a", nrow(DomainOnlyLandline)), rep("ab", nrow(DomainBothLandline)))
> FrameLandline <- cbind(FrameLandline, Domain)
>
> Domain <- c(rep("b", nrow(DomainOnlyCell)), rep("ba", nrow(DomainBothCell)))
> FrameCell <- cbind(FrameCell, Domain)
```

Now dual frame estimators, as PML estimator, can be computed:

```
> summary(PML(FrameLandline$Opinion, FrameCell$Opinion, FrameLandline$ProbLandline,
+   FrameCell$ProbCell, FrameLandline$Domain, FrameCell$Domain, N_A = 4982920,
+   N_B = 5707655))

Call:
PML(ysA = FrameLandline$Opinion, ysB = FrameCell$Opinion,
    pi_A = FrameLandline$ProbLandline, pi_B = FrameCell$ProbCell,
    domains_A = FrameLandline$Domain, domains_B = FrameCell$Domain,
    N_A = 4982920, N_B = 5707655)

Estimation:
              [,1]
Total 3.231325e+06
Mean  4.635634e-01

Variance Estimation:
```

```
                  [,1]
Var. Total 1.784362e+10
Var. Mean  3.672317e-04

Total Domain Estimations:
                  [,1]
Total dom. a   219145.1
Total dom. ab 2318841.9
Total dom. b  1346646.1
Total dom. ba 1457501.0

Mean Domain Estimations:
                 [,1]
Mean dom. a  0.4438149
Mean dom. ab 0.4990548
Mean dom. b  0.4172797
Mean dom. ba 0.4674919

Parameters:

gamma 0.3211534
```

As the overlap domain size is known, we can include additionally this information in the process and compute more accurate estimators as CalDF and CalSF.

```
> summary(CalDF(FrameLandline$Opinion, FrameCell$Opinion, FrameLandline$ProbLandline,
+   FrameCell$ProbCell, FrameLandline$Domain, FrameCell$Domain, N_A = 4982920,
+   N_B = 5707655, N_ab = 4339659))

Call:
CalDF(ysA = FrameLandline$Opinion, ysB = FrameCell$Opinion,
    pi_A = FrameLandline$ProbLandline, pi_B = FrameCell$ProbCell,
    domains_A = FrameLandline$Domain, domains_B = FrameCell$Domain,
    N_A = 4982920, N_B = 5707655, N_ab = 4339659)

Estimation:
             [,1]
Total 2.985028e+06
Mean  4.700153e-01

Variance Estimation:
                  [,1]
Var. Total 1.478990e+10
Var. Mean  3.666844e-04

Parameters:

eta 0.7296841
>
> summary(CalSF(FrameLandline$Opinion, FrameCell$Opinion, FrameLandline$ProbLandline,
+   FrameCell$ProbCell, FrameLandline$ProbCell, FrameCell$ProbLandline,
+   FrameLandline$Domain, FrameCell$Domain, N_A = 4982920, N_B = 5707655,
+   N_ab = 4339659))

Call:
CalSF(ysA = FrameLandline$Opinion, ysB = FrameCell$Opinion,
    pi_A = FrameLandline$ProbLandline, pi_B = FrameCell$ProbCell,
    pik_ab_B = FrameLandline$ProbCell, pik_ba_A = FrameCell$ProbLandline,
    domains_A = FrameLandline$Domain, domains_B = FrameCell$Domain, N_A = 4982920,
    N_B = 5707655, N_ab = 4339659)

Estimation:
             [,1]
Total 2.986787e+06
Mean  4.702923e-01
```

```
Variance Estimation:
                   [,1]
Var. Total 1.442969e+10
Var. Mean  3.577539e-04
```

Observe that the greater the information included in the estimation process is, the greater is the accuracy of the estimates.

Summary

The statistical literature about dual frame surveys started around 1960 and its development has evolved very quickly because these surveys are largely used by statistical agencies and private organizations to decrease sampling costs and to reduce frame undercoverage errors that could occur with the use of a single sampling frame.

Dual frame surveys can be more complicated to design and more complicated to analyze than those that use one frame only. There are several estimators of the population total available in the statistical literature. These estimators rely on weight adjustments to compensate for the multiplicity of the units in the overlap domain. Some of these estimators allow the handling of different types of auxiliary information at different levels. Nevertheless, none of the existing statistical software implements all of these estimators.

In this article we illustrate **Frames2**, a new R package for point and interval estimation in dual frame context. Functions in the package implement the most important estimators in the literature for population totals and means. We include two procedures (Pseudo-Empirical-Likelihood approach and calibration approach) to incorporate auxiliary information about frame sizes and also about one or several auxiliary variables in one or two frames. Post-stratification, raking ratio or regression estimation are all encompassed as particular cases of these estimation procedures. Additional functions for confidence interval estimation based on the jackknife variance estimation have been included as well.

The functionalities of the package **Frames2** have been illustrated using several data sets `DatA`, `DatB` and `Dat` (included in the package) corresponding to different complex surveys. We envision future additions to the package that will allow for extensions to more than two frames.

Finally, we would like to direct the reader to the package vignettes named `"estimation"` (*Estimation in a dual frame context*) and `"formatting.data"` (*Splitting and formatting data in a dual frame context*) for further examples and background information.

Acknowledgements

This study was partially supported by Ministerio de Educación, Cultura y Deporte (grant MTM2012-35650 and FPU grant program, Spain), by Consejería de Economía, Innovación, Ciencia y Empleo (grant SEJ2954, Junta de Andalucía, Spain) and under the support of the project PRIN-SURWEY (grant 2012F42NS8, Italy). The authors are grateful to Manuel Trujillo, (IESA, Institute of Advanced Social Studies) for providing data and information about the OPIA (Opinions and Attitudes of the Andalusian Population regarding Immigration) survey. This paper and software have been considerably improved by the comments of the editor and the reviewers.

Bibliography

A. Alfons, J. Holzer, and M. Templ. *laeken: Estimation of Indicators on Social Exclusion and Poverty*, 2014. URL `http://CRAN.R-project.org/package=laeken`. R package version 0.4.6. [p52]

A. Arcos, M. Rueda, M. G. Ranalli, and D. Molina. *Frames2: Estimation in Dual Frame Surveys*, 2015. URL `http://CRAN.R-project.org/package=Frames2`. R package version 0.1.1. [p53]

M. D. Bankier. Estimators based on several stratified samples with applications to multiple frame surveys. *Journal of the American Statistical Association*, 81(396):1074–1079, 1986. [p54, 55]

J. C. Deville. Estimation de la variance pour les enquêtes en deux phases. Manuscript, INSEE, Paris, 1993. [p56, 65]

W. A. Fuller and L. F. Burmeister. Estimation for samples selected from two overlapping frames. In *ASA Proceedings of the Social Statistics Sections*, pages 245–249, 1972. [p54, 55]

W. A. Fuller, W. Kennedy, D. Schell, G. Sullivan, and H. J. Park. *PCCARP*. Iowa State University Statistical Laboratory, 1989. [p52]

H. A. Gutierrez Rojas. *TeachingSampling: Selection of Samples and Parameter Estimation in Finite Population*, 2014. URL http://CRAN.R-project.org/package=TeachingSampling. R package version 3.2.1. [p52]

H. O. Hartley. Multiple frame surveys. In *Proceedings of the American Statistical Association, Social Statistics Sections*, pages 203–206, 1962. [p53, 54]

H. O. Hartley. Multiple frame methodology and selected applications. *Sankhyā C*, 36(3):99–118, 1974. [p54]

IBM Corporation. *IBM SPSS Statistics for Windows, Version 22.0*. Armonk, NY, 2013. URL http://www.ibm.com/software/analytics/spss/. [p52]

G. Kalton and D. W. Anderson. Sampling rare populations. *Journal of the Royal Statistical Society A*, 149 (1):65–82, 1986. [p54]

S. Lohr and J. Rao. Inference in dual frame surveys. *Journal of the American Statistical Association*, 95 (449):271–280, 2000. [p56]

T. Lumley. *survey: Analysis of Comples Survey Samples*, 2014. URL http://CRAN.R-project.org/package=survey. R package version 3.30. [p52]

M. H. Quenouille. Problems in plane sampling. *The Annals of Mathematical Statistics*, 20(3):355–375, 1949. [p56]

M. H. Quenouille. Notes on bias in estimation. *Biometrika*, 43(3/4):353–360, 1956. [p56]

M. G. Ranalli, A. Arcos, M. Rueda, and A. Teodoro. Calibration estimators in dual frames surveys. arXiv:1312.0761 [stat.ME], 2013. [p56]

J. N. K. Rao and C. J. Skinner. Estimation in dual frame surveys with complex designs. In *Proceedings of the Survey Method Section, Statistical Society of Canada*, pages 63–68, 1996. [p55]

J. N. K. Rao and C. Wu. Pseudo empirical likelihood inference for multiple frame surveys. *Journal of the American Statistical Association*, 105(492):1494–1503, 2010. [p55, 62]

Research Triangle Institute. *SUDAAN, Version 11.0.1*. Research Triangle Park, NC, 2013. URL http://www.rti.org/sudaan/. [p52]

SAS Institute Inc. *SAS Software, Version 9.4*. Cary, NC, 2013. URL http://www.sas.com/. [p52]

C. J. Skinner. On the efficiency of raking ratio estimation for multiple frame surveys. *Journal of the American Statistical Association*, 86(415):779–784, 1991. [p55]

C. J. Skinner and J. N. K. Rao. Estimation in dual frame surveys with complex designs. *Journal of the American Statistical Association*, 91(443):349–356, 1996. [p55]

Stata Corporation. *Stata Statistical Software, Version 14.0*. College Station, TX, 2015. URL http://www.stata.com/. [p52]

Systat Software Inc. *Systat, Version 13.0*. San Jose, California, 2009. URL http://www.systat.com. [p52]

M. Templ. *CRAN Task View: Official Statistics and Survey Methodology*, 2014. URL http://CRAN.R-project.org/view=OfficialStatistics. [p52]

Y. Tillé and A. Matei. *sampling: Survey Sampling*, 2012. URL http://CRAN.R-project.org/package=sampling. R package version 2.5. [p52, 64]

K. M. Wolter. *Introduction to Variance Estimation*. Springer, New York, 2nd edition, 2007. [p56]

C. Wu. Algorithms and R codes for the pseudo empirical likelihood method in survey sampling. *Survey Methodology*, 31(2):239–243, 2005. [p55, 64]

Antonio Arcos
Department of Statistics and Operational Research
University of Granada
Spain
arcos@ugr.es

David Molina
Department of Statistics and Operational Research
University of Granada
Spain
dmolinam@ugr.es

Maria Giovanna Rannalli
Department of Political Science
University of Perugia
Italy
giovanna.ranalli@stat.unipg.it

María del Mar Rueda
Department of Statistics and Operational Research
University of Granada
Spain
mrueda@ugr.es

The Complex Multivariate Gaussian Distribution

by Robin K. S. Hankin

Abstract Here I introduce package **cmvnorm**, a complex generalization of the **mvtnorm** package. A complex generalization of the Gaussian process is suggested and numerical results presented using the package. An application in the context of approximating the Weierstrass σ-function using a complex Gaussian process is given.

Introduction

Complex-valued random variables find applications in many areas of science such as signal processing (Kay, 1989), radio engineering (Ozarow, 1994), and atmospheric physics (Mandic et al., 2009). In this short paper I introduce **cmvnorm** (Hankin, 2015), a package for investigating one commonly encountered complex-valued probability distribution, the complex Gaussian.

The real multivariate Gaussian distribution is well supported in R by package **mvtnorm** (Genz et al., 2014), having density function

$$f(\mathbf{x};\mathbf{m},\Sigma) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{x}-\mathbf{m})}}{\sqrt{|2\pi\Sigma|}} \qquad \mathbf{x}\in\mathbb{R}^n, \tag{1}$$

where $|M|$ denotes the determinant of matrix M. Here, $\mathbf{m} = \mathbb{E}[\mathbf{X}] \in \mathbb{R}^n$ is the mean vector and $\Sigma = \mathbb{E}\left[(\mathbf{X}-\mathbf{m})(\mathbf{X}-\mathbf{m})^T\right]$ the covariance of random vector $\mathbf{X}$; we write $\mathbf{X} \sim \mathcal{N}_n(\mathbf{m},\Sigma)$. One natural generalization would be to consider $\mathbf{Z} \sim \mathcal{NC}_n(\mathbf{m},\Gamma)$, the complex multivariate Gaussian, with density function

$$f(\mathbf{z};\mathbf{m},\Gamma) = \frac{e^{-(\mathbf{z}-\mathbf{m})^*\Gamma^{-1}(\mathbf{z}-\mathbf{m})}}{|\pi\Gamma|} \qquad \mathbf{z}\in\mathbb{C}^n, \tag{2}$$

where $\mathbf{z}^*$ denotes the Hermitian transpose of complex vector $\mathbf{z}$. Now $\mathbf{m}\in\mathbb{C}^n$ is the complex mean and $\Gamma = \mathbb{E}\left[(\mathbf{Z}-\mathbf{m})(\mathbf{Z}-\mathbf{m})^*\right]$ is the complex variance; Γ is a Hermitian positive definite matrix. Note the simpler form of (2), essentially due to Gauss's integral operating more cleanly over the complex plane than the real line:

$$\int_{\mathbb{C}} e^{-z^*z}\,dz = \int_{x\in\mathbb{R}}\int_{y\in\mathbb{R}} e^{-(x^2+y^2)}\,dx\,dy = \int_{\theta=0}^{2\pi}\int_{r=0}^{\infty} e^{-r^2}r\,dr\,d\theta = \pi.$$

A zero mean complex random vector $\mathbf{Z}$ is said to be *circularly symmetric* (Goodman, 1963) if $\mathbb{E}\left[\mathbf{Z}\mathbf{Z}^T\right] = 0$, or equivalently $\mathbf{Z}$ and $e^{i\alpha}\mathbf{Z}$ have identical distributions for any $\alpha\in\mathbb{R}$. Equation (2) clearly has this property.

Most results from real multivariate analysis have a direct generalization to the complex case, as long as "transpose" is replaced by "Hermitian transpose". For example, $\mathbf{X} \sim \mathcal{N}_n(\mathbf{0},\Sigma)$ implies $B\mathbf{X} \sim \mathcal{N}_n\left(\mathbf{0}, B^T\Sigma B\right)$ for any constant matrix $B\in\mathbb{R}^{m\times n}$, and analogously $\mathbf{Z} \sim \mathcal{NC}_n(\mathbf{0},\Gamma)$ implies $B\mathbf{Z} \sim \mathcal{NC}_n(\mathbf{0}, B^*\Gamma B)$, $B\in\mathbb{C}^{m\times n}$. Similar generalizations operate for Schur complement methods on partitioned matrices.

Also, linear regression generalizes similarly. Specifically, consider $\mathbf{y}\in\mathbb{R}^n$. If $\mathbf{y} = X\boldsymbol{\beta}+\boldsymbol{\epsilon}$ where X is a $n\times p$ design matrix, $\boldsymbol{\beta}\in\mathbb{R}^p$ a vector of regression coefficients and $\boldsymbol{\epsilon} \sim \mathcal{N}_n(\mathbf{0},\Sigma)$ is a vector of errors, then $\hat{\boldsymbol{\beta}} = \left(X^T\Sigma^{-1}X\right)^{-1}X^T\Sigma^{-1}\mathbf{y}$ is the maximum likelihood estimator for $\boldsymbol{\beta}$. The complex generalization is to write $\mathbf{z} = Z\boldsymbol{\beta}+\boldsymbol{\epsilon}$, $Z\in\mathbb{C}^{n\times p}$, $\boldsymbol{\beta}\in\mathbb{C}^p$, $\boldsymbol{\epsilon} \sim \mathcal{NC}_n(\mathbf{0},\Gamma)$ which gives $\hat{\boldsymbol{\beta}} = \left(Z^*\Gamma^{-1}Z\right)^{-1}Z^*\Gamma^{-1}\mathbf{z}$. Such considerations suggest a natural complex generalization of the Gaussian process.

This short vignette introduces the **cmvnorm** package which furnishes some functionality for the complex multivariate Gaussian distribution, and applies it in the context of a complex generalization of the **emulator** package (Hankin, 2005), which implements functionality for investigating (real) Gaussian processes.

The package in use

Random complex vectors are generated using the rcmvnorm() function, analogous to rmvnorm():

```
> set.seed(1)
> library("cmvnorm", quietly = TRUE)
> cm <- c(1, 1i)
> cv <- matrix(c(2, 1i, -1i, 2), 2, 2)
> (z <- rcmvnorm(6, mean = cm, sigma = cv))

                     [,1]                  [,2]
[1,] 0.9680986+0.5525419i  0.0165969+2.9770976i
[2,] 0.2044744-1.4994889i  1.8320765+0.8271259i
[3,] 1.0739973+0.2279914i -0.7967020+0.1736071i
[4,] 1.3171073-0.9843313i  0.9257146+0.5524913i
[5,] 1.3537303-0.8086236i -0.0571337+0.3935375i
[6,] 2.9751506-0.1729231i  0.3958585+3.3128439i
```

Function dcmvnorm() returns the density according to (2):

```
> dcmvnorm(z, cm, cv)

[1] 5.103754e-04 1.809636e-05 2.981718e-03 1.172242e-03 4.466836e-03 6.803356e-07
```

So it is possible to determine a maximum likelihood estimate for the mean using direct numerical optimization

```
> helper <- function(x) c(x[1] + 1i * x[2], x[3] + 1i * x[4])
> objective <- function(x, cv)
+   -sum(dcmvnorm(z, mean = helper(x), sigma = cv, log = TRUE))
> helper(optim(c(1, 0, 1, 0), objective, cv = cv)$par)

[1] 1.315409-0.447863i 0.385704+1.372762i
```

(helper functions are needed because optim() optimizes over $\mathbb{R}^n$ as opposed to $\mathbb{C}^n$). This shows reasonable agreement with the true value of the mean and indeed the analytic value of the MLE, specifically

```
> colMeans(z)

[1] 1.315426-0.447472i 0.386068+1.372784i
```

The Gaussian process

In the context of the emulator, a (real) Gaussian process is usually defined as a random function $\eta\colon \mathbb{R}^p \longrightarrow \mathbb{R}$ which, for any set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in its domain $\mathcal{D}$ the random vector $\{\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)\}$ is multivariate Gaussian.

It is convenient to specify $\mathbb{E}[\eta(\mathbf{x})|\boldsymbol{\beta}] = h(\mathbf{x})\boldsymbol{\beta}$, that is, the expectation of the process at point $\mathbf{x} \in \mathcal{D}$ conditional on the (unknown) vector of coefficients $\boldsymbol{\beta}$. Here $h\colon \mathbb{R}^p \longrightarrow \mathbb{R}^q$ specifies the q known regressor functions of $\mathbf{x} = (x_1, \ldots, x_p)^T$; a common choice is $h(\mathbf{x}) = (1, x_1, \ldots, x_p)^T$ [giving $q = p + 1$], but one is in principle free to choose any function of $\mathbf{x}$. One writes $H^T = (h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n))$ when considering the entire design matrix X; the R idiom is regressor.multi().

The covariance is typically given by

$$\operatorname{cov}(\eta(\mathbf{x}), \eta(\mathbf{x}')) = V(\mathbf{x} - \mathbf{x}'),$$

where $V\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ must be chosen so that the variance matrix of any finite set of observations is always positive-definite. Bochner's theorem (Feller, 1971, chapter XIX) shows that $V(\cdot)$ must be proportional to the characteristic function (CF) of a symmetric probability Borel measure.

Oakley (1999) uses techniques which have clear complex analogues to show that the posterior mean of $\eta(\mathbf{x})$ is given by

$$h(\mathbf{x})^T \boldsymbol{\beta} + (\operatorname{cov}(\mathbf{x}, \mathbf{x}_1), \ldots, \operatorname{cov}(\mathbf{x}, \mathbf{x}_n))^T A^{-1}\left(\mathbf{y} - H\hat{\boldsymbol{\beta}}\right). \tag{3}$$

Here A is an $n \times n$ matrix of correlations between the observations, $\sigma^2 A_{ij} = \operatorname{cov}\left(\eta\left(\mathbf{x}_i\right), \eta\left(\mathbf{x}_j\right)\right)$ where σ^2 is an overall variance term; and $\hat{\boldsymbol{\beta}} = \left(X^T A^{-1} X\right)^{-1} X^T A^{-1}\mathbf{y}$ is the maximum likelihood estimator for $\boldsymbol{\beta}$.

Equation (3) furnishes a cheap approximation to $\eta\left(\mathbf{x}\right)$ and is known as the 'emulator'.

Complex Gaussian processes

The complex case is directly analogous, with $\eta\colon \mathbb{C}^p \longrightarrow \mathbb{C}$ and $\boldsymbol{\beta} \in \mathbb{C}^q$. Writing $\operatorname{cov}\left(\eta\left(\mathbf{z}_1\right), \ldots, \eta\left(\mathbf{z}_n\right)\right) = \Omega$, so that element (i,j) of matrix Ω is $\operatorname{cov}\left(\eta\left(\mathbf{z}_i\right), \eta\left(\mathbf{z}_j\right)\right)$, we may relax the requirement that Ω be symmetric positive definite to requiring only Hermitian positive definiteness. This allows one to use the characteristic function of *any*, possibly non-symmetric, random variable Ψ with density function $f\colon \mathbb{R}^p \longrightarrow \mathbb{R}$ and characteristic function ϕ:

$$\Omega_{ij} = \operatorname{cov}\left(\eta\left(\mathbf{z}_i\right), \eta\left(\mathbf{z}_j\right)\right) = \phi\left(\mathbf{z}_i - \mathbf{z}_j\right). \tag{4}$$

That Ω remains Hermitian positive definite may be shown by evaluating a quadratic form with it and arbitrary $\mathbf{w} \in \mathbb{C}^n$ and establishing that it is real and non-negative:

$$\begin{aligned}
\mathbf{w}^*\Omega\mathbf{w} &= \sum_{i,j} \overline{\mathbf{w}_i} \operatorname{cov}\left(\eta\left(\mathbf{z}_i\right), \eta\left(\mathbf{z}_j\right)\right)\mathbf{w}_j && \text{definition of quadratic form}\\
&= \sum_{i,j} \overline{\mathbf{w}_i}\phi\left(\mathbf{z}_i - \mathbf{z}_j\right)\mathbf{w}_j && \text{covariance function is the CF of } \Psi\\
&= \sum_{i,j} \overline{\mathbf{w}_i}\left[\int_{\mathbf{t}\in\mathbb{C}^n} e^{\mathrm{i}\operatorname{Re}\mathbf{t}^*(\mathbf{z}_i - \mathbf{z}_j)} f\left(\mathbf{t}\right)\,d\mathbf{t}\right]\mathbf{w}_j && \text{definition of CF of } \Psi\\
&= \int_{\mathbf{t}\in\mathbb{C}^n}\left[\sum_{i,j}\overline{\mathbf{w}_i} e^{\mathrm{i}\operatorname{Re}\mathbf{t}^*(\mathbf{z}_i - \mathbf{z}_j)}\mathbf{w}_j f\left(\mathbf{t}\right)\right]\,d\mathbf{t} && \text{integration and summation commute}\\
&= \int_{\mathbf{t}\in\mathbb{C}^n}\left[\sum_{i,j}\overline{\mathbf{w}_i} e^{\mathrm{i}\operatorname{Re}(\mathbf{t}^*\mathbf{z}_i)}\overline{\overline{\mathbf{w}_j} e^{\mathrm{i}\operatorname{Re}(\mathbf{t}^*\mathbf{z}_j)}} f\left(\mathbf{t}\right)\right]\,d\mathbf{t} && \text{expand and rearrange}\\
&= \int_{\mathbf{t}\in\mathbb{C}^n}\left|\sum_{i}\overline{\mathbf{w}_i} e^{\mathrm{i}\operatorname{Re}(\mathbf{t}^*\mathbf{z}_i)}\right|^2 f\left(\mathbf{t}\right)\,d\mathbf{t} && \text{algebra}\\
&\geqslant 0. && \text{integral of sum of real positive functions}
\end{aligned}$$

(This motivates the definition of the characteristic function of a complex multivariate random variable $\mathbf{Z}$ as $\mathbb{E}\left[e^{i\operatorname{Re}(\mathbf{t}^*\mathbf{Z})}\right]$). Thus the covariance matrix is Hermitian positive definite: although its entries are not necessarily real, its eigenvalues are all nonnegative.

In the real case one typically chooses Ψ to be a zero-mean Gaussian distribution; in the complex case one can use the complex multivariate distribution given in equation (2) which has characteristic function

$$\exp\left(i\operatorname{Re}\left(\mathbf{t}^*\mathbf{m}\right) - \frac{1}{4}\mathbf{t}^*\Gamma\mathbf{t}\right) \tag{5}$$

and following Hankin (2012) in writing $\mathfrak{B} = \Gamma/4$, we can write the variance matrix as a product of a (real) scalar σ^2 term and

$$c\left(\mathbf{t}\right) = \exp\left(i\operatorname{Re}\left(\mathbf{t}^*\mathbf{m}\right) - \mathbf{t}^*\mathfrak{B}\mathbf{t}\right). \tag{6}$$

Thus the covariance matrix Ω is given by

$$\Omega_{ij} = \operatorname{cov}\left(\eta\left(\mathbf{z}_i\right), \eta\left(\mathbf{z}_j\right)\right) = \sigma^2 c\left(\mathbf{z}_i - \mathbf{z}_j\right). \tag{7}$$

In (6), $\mathfrak{B}$ has the same meaning as in conventional emulator techniques and controls the modulus of the covariance between $\eta\left(\mathbf{z}\right)$ and $\eta\left(\mathbf{z}'\right)$; $\mathbf{m}$ governs the phase.

Given the above, it seems to be reasonable to follow Oakley (1999) and admit only diagonal $\mathfrak{B}$; but now distributions with nonzero mean can be considered (compare the real case which requires a zero mean). A parametrization using diagonal $\mathfrak{B}$ and complex mean vector requires $3p$ (real) hyperparameters; compare $2p$ if $\mathbb{C}^p$ is identified with $\mathbb{R}^{2p}$.

Functions of several complex variables

Analytic functions of several complex variables are an important and interesting class of objects; Krantz (1987) motivates and discusses the discipline. Formally, consider $f\colon \mathbb{C}^n \longrightarrow \mathbb{C}$, $n \geqslant 2$ and write $f(z_1, \ldots, z_n)$. Function f is *analytic* if it satisfies the Cauchy-Riemann conditions in each variable separately, that is $\partial f/\partial\overline{z}_j = 0$, $1 \leqslant j \leqslant n$.

Such an f is continuous (due to a "non-trivial theorem of Hartogs") and continuously differentiable to arbitrarily high order. Krantz goes on to state some results which are startling if one's exposure to complex analysis is restricted to functions of a single variable: for example, any isolated singularity is removable.

Numerical illustration of these ideas

The natural definition of complex Gaussian processes above, together with the features of analytic functions of several complex variables, suggests that a complex emulation of analytic functions of several complex variables might be a useful technique.

The ideas presented above, and the **cmvnorm** package, can now be used to sample directly from an appropriate complex Gaussian distribution and estimate the roughness parameters:

```
> val <- latin.hypercube(40, 2, names = c("a", "b"), complex = TRUE)
> head(val)

                   a                b
[1,] 0.7375+0.2375i 0.2375+0.7125i
[2,] 0.6875+0.5875i 0.1375+0.3375i
[3,] 0.4625+0.5375i 0.9875+0.5875i
[4,] 0.7875+0.0625i 0.0625+0.7875i
[5,] 0.3875+0.0375i 0.5875+0.7625i
[6,] 0.2125+0.5625i 0.7625+0.9625i
```

(function `latin.hypercube()` is used to generate a random complex design matrix). We may now specify a variance matrix using simple values for the roughness hyperparameters $\mathfrak{B} = \left(\begin{smallmatrix}1&0\\0&2\end{smallmatrix}\right)$ and $\mathbf{m} = (1, i)^T$:

```
> true_scales <- c(1, 2)
> true_means <- c(1, 1i)
> A <- corr_complex(val, means = true_means, scales = true_scales)
> round(A[1:4, 1:4], 2)

           [,1]       [,2]       [,3]       [,4]
[1,] 1.00+0.00i 0.59-0.27i 0.25-0.10i 0.89+0.11i
[2,] 0.59+0.27i 1.00+0.00i 0.20+0.00i 0.42+0.26i
[3,] 0.25+0.10i 0.20+0.00i 1.00+0.00i 0.10+0.06i
[4,] 0.89-0.11i 0.42-0.26i 0.10-0.06i 1.00+0.00i
```

Function `corr_complex()` is a complex generalization of `corr()`; matrix `A` is Hermitian positive-definite:

```
> all(eigen(A)$values > 0)

[1] TRUE
```

It is now possible to make a single multivariate observation d of this process, using $\boldsymbol{\beta} = (1, 1+i, 1-2i)^T$:

```
> true_beta <- c(1, 1+1i, 1-2i)
> d <- drop(rcmvnorm(n = 1, mean = regressor.multi(val) %*% true_beta, sigma = A))
> head(d)

[1] 3.212719+1.594901i 1.874278+0.345517i 3.008503-0.767618i 3.766526+2.071882i
[5] 3.712913+0.800983i 3.944167+0.924833i
```

Thus d is a single observation from a complex multivariate Gaussian distribution. Most of the functions of the **emulator** package operate without modification. Thus `betahat.fun()`, which calculates the maximum likelihood estimate $\hat{\boldsymbol{\beta}} = \left(H^* A^{-1} H\right)^{-1} H^* A^{-1}\mathbf{y}$ takes complex values directly:

```
> betahat.fun(val, solve(A), d)

              const                   a                   b
0.593632-0.0128655i 0.843608+1.0920437i 1.140372-2.5053751i
```

However, because the likelihood function is different, the `interpolant()` functionality is implemented in the **cmvnorm** package by `interpolant.quick.complex()`, named in analogy to function `interpolant.quick()` of package **emulator**.

For example, it is possible to evaluate the posterior distribution of the process at $(0.5, 0.3 + 0.1i)$, a point at which no observation has been made:

```
> interpolant.quick.complex(rbind(c(0.5, 0.3+0.1i)), d, val, solve(A),
+   scales = true_scales, means = true_means, give.Z = TRUE)

$mstar.star
[1] 1.706402-1.008601i

$Z
[1] 0.203295

$prior
[1] 1.608085-0.104419i
```

Thus the posterior distribution for the process is complex Gaussian at this point with a mean of about $1.71 - 1.01i$ and a variance of about 0.2.

Analytic functions

These techniques are now used to emulate an analytic function of several complex variables. A complex function's being analytic is a very strong restriction; Needham (2004) uses 'rigidity' to describe the severe constraint that analyticity represents.

Here the Weierstrass σ-function (Chandrasekharan, 1985) is chosen as an example, on the grounds that Littlewood and Offord (1948) consider it to be a typical entire function in a well-defined sense. The elliptic package (Hankin, 2006) is used for numerical evaluation.

The σ-function takes a primary argument z and two invariants g_1, g_2, so a three-column complex design matrix is required:

```
> library("elliptic")
> valsigma <- 2 + 1i + round(latin.hypercube(30, 3,
+   names = c("z", "g1", "g2"), complex = TRUE)/4, 2)
> head(valsigma)

              z         g1         g2
[1,] 2.17+1.15i 2.09+1.22i 2.21+1.09i
[2,] 2.11+1.01i 2.04+1.03i 2.25+1.15i
[3,] 2.10+1.04i 2.15+1.00i 2.22+1.20i
[4,] 2.13+1.10i 2.24+1.21i 2.01+1.16i
[5,] 2.20+1.00i 2.20+1.06i 2.08+1.08i
[6,] 2.05+1.10i 2.19+1.04i 2.11+1.03i
```

(an offset is needed because $\sigma(z, g_1, g_2) = z + \mathcal{O}(z^5)$). The σ-function can now be evaluated at the points of the design matrix:

```
> dsigma <- apply(valsigma, 1, function(u) sigma(u[1], g = u[2:3]))
```

One way of estimating the roughness parameters is to use maximum likelihood. The likelihood for any set of roughness parameters is given by Oakley (1999) as $\left(\sigma^2\right)^{-\frac{n-q}{2}} |A|^{-1/2} \left|H^T A^{-1} H\right|^{-1/2}$ with complex generalization $\left(\sigma^2\right)^{-(n-q)} |A|^{-1} \left|H^* A^{-1} H\right|^{-1}$ which is calculated in the package by function `scales.likelihood.complex()`; this can be used to return the log-likelihood for a specific set of roughness parameters:

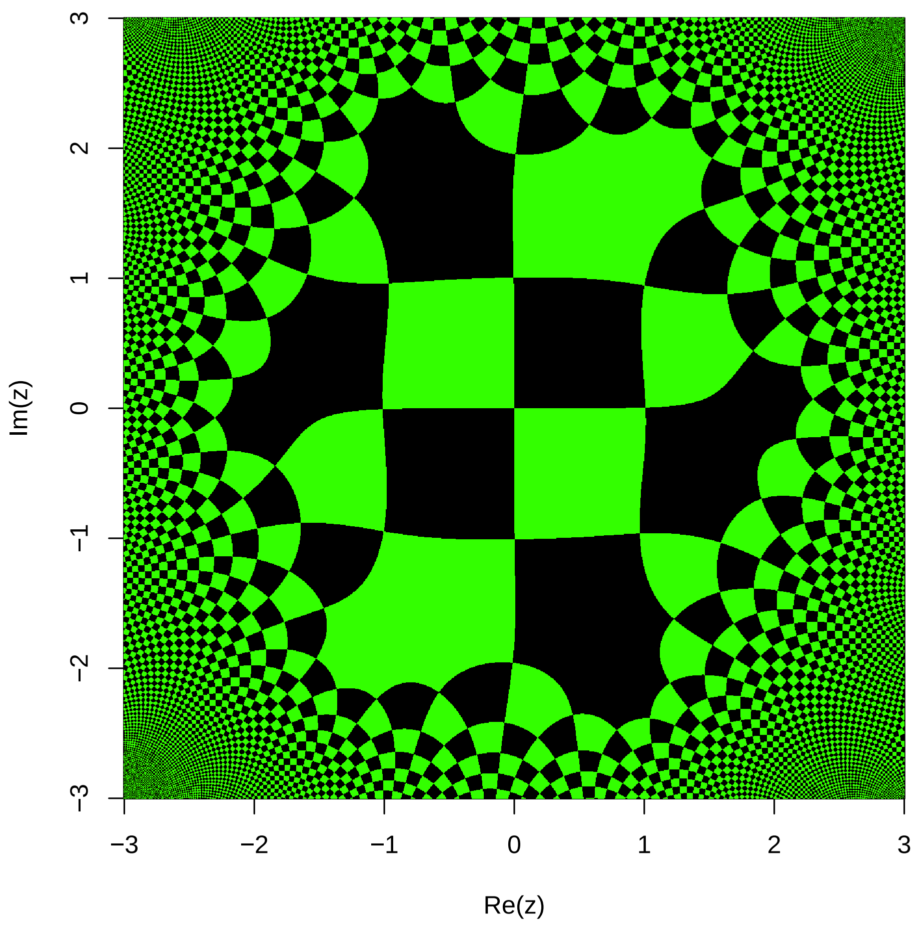


Figure 1: Visualization of the Weierstrass σ-function, specifically $\sigma\left(z; 2+i, 2.2+1.1i\right)$ in the region of the complex plane $-4 \leqslant \operatorname{Re}\left(z\right), \operatorname{Im}\left(z\right) \leqslant +4$; visualization is scheme 13 of Hankin (2006).

```
> scales.likelihood.complex(scales = c(1, 1, 2), means = c(1, 1+1i, 1-2i),
+   zold = valsigma, z = dsigma, give_log = TRUE)

[1] 144.5415
```

Numerical methods can then be used to find the maximum likelihood estimate. Because function `optim()` optimizes over $\mathbb{R}^n$, helper functions are again needed which translate from the optimand to scales and means:

```
> scales <- function(x) exp(x[c(1, 2, 2)])
> means <-  function(x) x[c(3, 4, 4)] + 1i * x[c(5, 6, 6)]
```

Because the diagonal elements of $\mathfrak{B}$ are strictly positive, their *logarithms* are optimized, following Hankin (2005); it is implicitly assumed that the scales and means associated with g_1 and g_2 are equal.

```
> objective <- function(x, valsigma, dsigma)
+   -scales.likelihood.complex(scales = scales(x), means = means(x),
+     zold = valsigma, z = dsigma)
> start <- c(-0.538, -5.668, 0.6633, -0.0084, -1.73, -0.028)
> jj <- optim(start, objective, valsigma = valsigma,  dsigma = dsigma,
+   method = "SANN", control = list(maxit = 100))
> (u <- jj$par)

[1] -0.5380 -5.6680  0.6633 -0.0084 -1.7300 -0.0280
```

Function `corr_complex()` may now be used to calculate the covariance of the observations:

```
> Asigma <- corr_complex(z1 = valsigma, scales = scales(u), means = means(u))
```

So now we can compare the emulator against the "true" value:

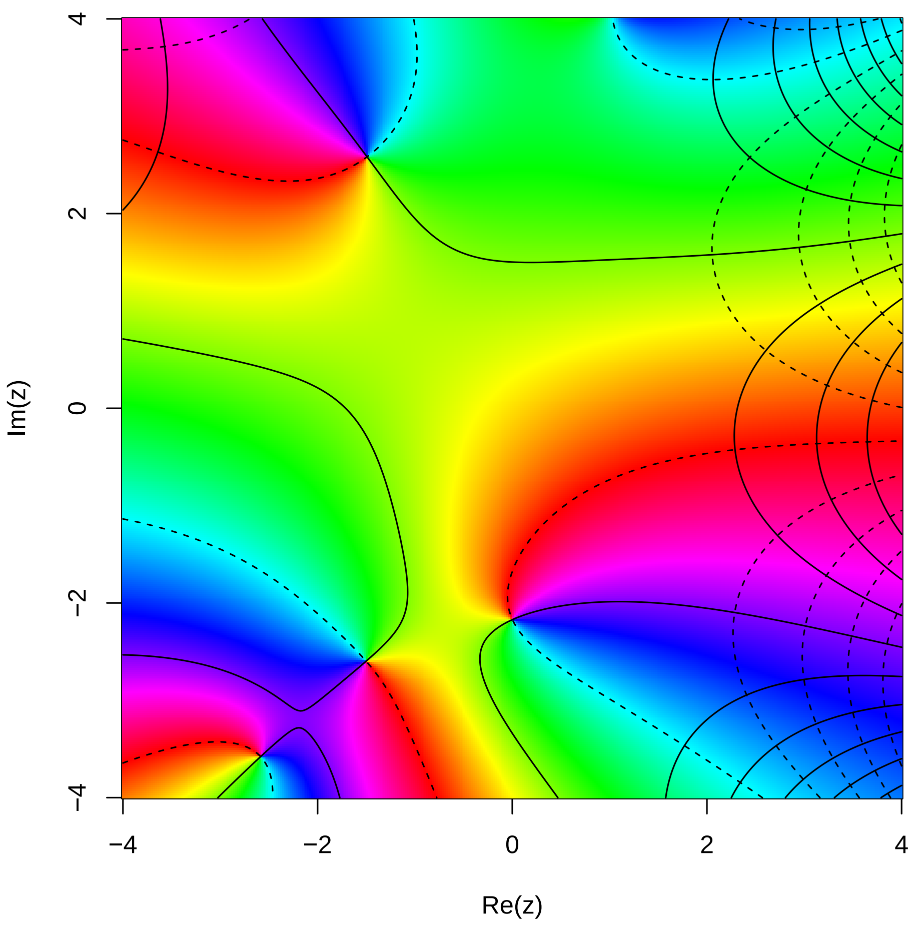


Figure 2: Visualization of the Weierstrass σ-function, specifically $\sigma\left(6+1i; g_1 = z, g_2 = 1\right)$ in the region of the complex plane $-4 \leqslant \mathrm{Re}\left(z\right), \mathrm{Im}\left(z\right) \leqslant +4$; visualization is scheme 8 of Hankin (2006).

```
> interpolant.quick.complex(rbind(c(2+1i, 2+1i, 2+1i)), zold = valsigma,
+   d = dsigma, Ainv = solve(Asigma), scales = scales(u), means = means(u))

[1] 3.078956+1.259993i

> sigma(2 + 1i, g = c(2 + 1i, 2 + 1i))

[1] 3.078255+1.257819i
```

showing reasonable agreement. It is also possible to test the hypothesis $H_{\mathbb{R}}: \mathbf{m} \in \mathbb{R}^2$ (that is, the variance matrix A is real), by calculating the likelihood ratio of the unconstrained model (6) to that obtained by $H_{\mathbb{R}}$. This may be achieved by constraining the optimization to satisfy $\mathbf{m} \in \mathbb{R}^2$:

```
> ob2 <- function(x, valsigma, dsigma)
+   -scales.likelihood.complex(scales = scales(x), means = c(0, 0, 0),
+     zold = valsigma, z = dsigma)
> jjr <- optim(u[1:2], ob2, method = "SANN", control = list(maxit = 1000),
+   valsigma = valsigma, dsigma = dsigma)
> (ur <- jjr$par)

[1]  0.2136577 -4.2640825
```

so the test statistic D is given by

```
> LR <- scales.likelihood.complex(scales = scales(ur), means = c(0, 0, 0),
+   zold = valsigma, z = dsigma)
> LC <- scales.likelihood.complex(scales = scales(u), means = means(u),
+   zold = valsigma, z = dsigma)
> (D <- 2 * (LC - LR))
```

```
[1] 22.17611
```

Observing that D is in the tail region of its asymptotic distribution, χ^2_3, the hypothesis $H_{\mathbb{R}}$ may be rejected.

Conclusions

The **cmvnorm** package for the complex multivariate Gaussian distribution has been introduced and motivated. The Gaussian process has been generalized to the complex case, and a complex generalization of the emulator technique has been applied to an analytic function of several complex variables. The complex variance matrix was specified using a novel parameterization which accommodated non-real covariances in the context of circulary symmetric random variables. Further work might include numerical support for the complex multivariate Student t distribution.

Bibliography

K. Chandrasekharan. *Elliptic Functions*. Springer-Verlag, 1985. [p77]

W. Feller. *An Introduction to Probability Theory and its Applications*, volume 2. Wiley, 1971. [p74]

A. Genz, F. Bretz, T. Miwa, X. Mi, F. Leisch, F. Scheipl, and T. Hothorn. *mvtnorm: Multivariate Normal and t Distributions*, 2014. URL http://CRAN.R-project.org/package=mvtnorm. R package version 1.0-0. [p73]

N. R. Goodman. Statistical analysis based on a certain multivariate complex Gaussian distribution (an introduction). *The Annals of Mathematical Statistics*, 34(1):152–177, Mar. 1963. [p73]

R. K. S. Hankin. Introducing BACCO, an R bundle for Bayesian analysis of computer code output. *Journal of Statistical Software*, 14(16):1–20, Oct. 2005. [p73, 78]

R. K. S. Hankin. Introducing elliptic, an R package for elliptic and modular functions. *Journal of Statistical Software*, 15(7):1–22, Feb. 2006. [p77, 78, 79]

R. K. S. Hankin. Introducing multivator: A multivariate emulator. *Journal of Statistical Software*, 46(8): 1–20, Feb. 2012. [p75]

R. K. S. Hankin. *cmvnorm: The Complex Multivariate Gaussian Distribution*, 2015. URL http://CRAN.R-project.org/package=cmvnorm. R package version 1.0-2. [p73]

S. Kay. *Modern Spectral Analysis*. Prentice-Hall, 1989. [p73]

S. G. Krantz. What is several complex variables? *The American Mathematical Monthly*, 94(3):236–256, Mar. 1987. [p76]

J. E. Littlewood and A. C. Offord. On the distribution of zeros and a-values of a random integral function (II). *The Annals of Mathematics*, 49(4):885–952, Oct. 1948. [p77]

D. P. Mandic, S. Javidi, S. L. Goh, A. Kuh, and K. Aihara. Complex-valued prediction of wind profile using augmented complex statistics. *Renewable Energy*, 34:196–201, 2009. [p73]

T. Needham. *Visual Complex Analysis*. Clarendon Press, Oxford, 2004. [p77]

J. Oakley. *Bayesian Uncertainty Analysis for Complex Computer Codes*. PhD thesis, University of Sheffield, 1999. [p74, 75, 77]

L. H. Ozarow. Information theoretic considerations for cellular mobile radio. *IEEE Transactions on Vehicular Technology*, 43(2):359–378, May 1994. [p73]

Robin K. S. Hankin
Auckland University of Technology
2-14 Wakefield Street
Auckland NZ
hankin.robin@gmail.com

sae: An R Package for Small Area Estimation

by Isabel Molina and Yolanda Marhuenda

Abstract We describe the R package **sae** for small area estimation. This package can be used to obtain model-based estimates for small areas based on a variety of models at the area and unit levels, along with basic direct and indirect estimates. Mean squared errors are estimated by analytical approximations in simple models and applying bootstrap procedures in more complex models. We describe the package functions and show how to use them through examples.

The R package at a glance

The R package **sae** implements small area estimation methods under the following area-level models:

- Fay-Herriot model (including common fitting methods);
- extended Fay-Herriot model that accounts for spatial correlation;
- extended Fay-Herriot model allowing for spatio-temporal correlation.

The package also includes small area estimation methods based on the basic unit level model called the nested-error linear regression model. The available estimation methods under this model are:

- Empirical best linear unbiased predictors (EBLUPs) of area means under the nested-error linear regression model for the target variable.
- Empirical Best/Bayes (EB) estimates of general nonlinear area parameters under the nested-error linear regression model for Box-Cox or power transformations of the target variable.

Methods for estimation of the corresponding uncertainty measures of the small area estimators obtained from the above models are also included. Additionally, the package includes the following basic direct and indirect estimators

- Direct Horvitz-Thompson estimators of small area means under general sampling designs;
- Post-stratified synthetic estimator;
- Composite estimator.

This paper describes the above model-based small area estimation techniques and illustrates the use of the corresponding functions through suitable examples. For a description of the direct and basic indirect estimators included in the package and a detailed description of all implemented methodology, see `http://CRAN.R-project.org/package=sae`.

Introduction

The growing demand for more timely and detailed information, together with the high cost of interviews often leads to an extensive exploitation of survey data. Indeed, many times survey data are used to produce estimates in smaller domains or areas than those for which the survey was originally planned. For an area with a small sample size, a direct estimator, based only on the sample data coming from that area, might be very unreliable. This sample size limitation prevents the production of statistical figures at the requested level and therefore restricts the availability of statistical information for the public or the particular user. In contrast, an indirect estimator for an area also uses external data from other areas so as to increase efficiency by increasing the effective sample size. Among indirect estimators, we find those based on explicit regression models, called model-based estimators. These estimators are based on assuming a relation between the target variable and some explanatory variables that is constant across areas. The common model parameters are estimated using the whole bunch of sample data, which often leads to small area estimators with appreciably better efficiency than direct estimators as long as model assumptions hold. Thus, these techniques provide statistical figures at a very disaggregated level without increasing the area-specific sample sizes and therefore without increasing the survey cost. The small area estimation (SAE) methods included in the R package **sae** have applications in many different fields such as official statistics, agriculture, ecology, medicine and engineering. For a comprehensive account of SAE techniques, see Rao (2003).

The R package **sae** is mainly designed for model-based small area estimation. Nevertheless, simple direct and indirect estimators are included for didactic purposes and to allow the user to do cross comparisons between the very simple indirect methods and the more advanced model-based methods. Model-based point estimators can be supplemented with their corresponding estimated mean squared errors (MSEs), which are computed using analytical approximations in some cases and bootstrap procedures in other cases.

Area level models are used to obtain small area estimators when auxiliary data are available only as area aggregates. The basic area level model is the Fay-Herriot (FH) model (Fay and Herriot, 1979). Small area estimates based on this model and analytical MSE estimates can be obtained using the functions `eblupFH()` and `mseFH()` respectively.

An extension of the basic FH model to the case of (unexplained) spatial correlation among data from neighboring areas is the spatial Fay-Herriot (SFH) model. The function `eblupSFH` considers the SFH model in which area effects are assumed to follow a simultaneous autoregressive process of order one or SAR(1) process. Small area estimates supplemented with analytical MSE estimates can be obtained using the function `mseSFH()`. Alternatively, parametric and non-parametric bootstrap MSE estimates for the small area estimators obtained from the SFH model are given by the functions `pbmseSFH()` and `npbmseSFH()` respectively.

A spatio-temporal Fay-Herriot (STFH) model can be used when data from several periods of time are available and there is also spatial correlation. Apart from the area effects following a SAR(1) process, the STFH model considered by function `eblupSTFH()` includes time effects nested within areas, following for each domain an i.i.d. autorregresive process of order 1 or AR(1). The function `pbmseSTFH()` gives small area estimates and parametric bootstrap MSE estimates.

When auxiliary information is available at the unit level, the basic small area estimators are those based on the nested error linear regression model of Battese et al. (1988), called hereafter BHF model. Function `eblupBHF()` gives estimates of small area means based on BHF model. Parametric bootstrap MSE estimates are obtained calling function `pbmseBHF()`.

General small area parameters obtained as a nonlinear function of the response variable in the model, such as income-based poverty indicators, can be estimated under BHF model using function `ebBHF()`. Function `pbmseebBHF()` gives the corresponding parametric bootstrap MSE estimates.

The paper is structured as follows. First, we discuss the differences between design and model based inference and introduce the notation used throughout the paper. Then, we describe one by one the model-based SAE methods implemented in the package. For each method, we briefly describe the theory behind and the use of the functions, including suitable examples. Finally, we summarize other existing software for small area estimation.

Design versus model-based inference

In survey sampling, the population is a finite collection of distinguishable and countable units. The measurements of the target variable in the population units are assumed to be non-stochastic and the aim is to estimate characteristics of the population, i.e., functions of the population measurements of the study variable in the population units, which are consequently non-stochastic as well. A sample is simply a collection of population units and inference is typically carried out under the probability distribution induced by the random mechanism used to draw the sample, i.e., the sampling design. Thus, desirable properties of estimators such as unbiasedness are established in terms of averages over all possible samples.

In model-based inference, the term population refers simply to a random variable and, in the simplest case, the sample is a collection of independent variables distributed identically as the original random variable. The parameters of interest are characteristics of the probability distribution of the original random variable such as moments, which are assumed to be fixed under the frequentist setup.

In small area estimation, the subpopulations of interest are called indistinctly areas or domains. These areas are assumed to be finite although they are typically large. However, due to the lack of sample data within those areas, models are needed to link all areas through some common parameters so as to “borrow strength" from related areas and then to improve efficiency as long as model assumptions hold. Thus, model-based small area methods combine the finite population setup with the randomness of the measurements of the variable of interest in the population units, which are assumed to follow a regression model. Consequently, target quantities, defined as functions of the population measurements, are also random.

Notation

As mentioned above, here we consider a large but finite population U. This population is assumed to be partitioned into D mutually exclusive and exhaustive domains or areas $U_1, \ldots, U_D$ of sizes

$N_1, \ldots, N_D$. Let Y_{dj} be the measurement of the variable of interest for individual j within area d and let $\mathbf{y}_d = (Y_{d1}, \ldots, Y_{dN_d})^\top$ be the vector of measurements for area d. The target parameters have the form $\delta_d = h(\mathbf{y}_d)$, $d = 1, \ldots, D$, for a known measurable function h. Particular target parameters of common interest are the domain means

$$\delta_d = \bar{Y}_d = N_d^{-1} \sum_{j=1}^{N_d} Y_{dj}, \quad d = 1, \ldots, D.$$

Estimation of the target parameters is based on a sample s drawn from the population U. Let s_d be the subsample from domain U_d of size n_d, $d = 1, \ldots, D$, where $n = \sum_{d=1}^{D} n_d$ is the total sample size. We will denote by $r_d = U_d - s_d$ the sample complement from domain d of size $N_d - n_d$, for $d = 1, \ldots, D$.

Estimation of the area parameters $\delta_d = h(\mathbf{y}_d)$, $d = 1, \ldots, D$, can be done using area or unit-level models. In area level models, the auxiliary information comes in the form of aggregated values of some explanatory variables at the domains, typically true area means. In contrast, unit-level models make use of the individual values of the explanatory variables.

The package **sae** contains functions that provide small area estimators under both types of models. Functions for point estimation based on area level models include `eblupFH()`, `eblupSFH()` and `eblupSTFH()`. Functions for unit-level data are `eblupBHF()` and `ebBHF()`. Functions for estimation of the usual accuracy measures are also included. Below we describe the assumed models and the use of these functions, including examples of use. The package **sae** depends on packages **nlme** (Pinheiro et al., 2013) and **MASS** (Venables and Ripley, 2002). The examples of these functions have been run under R version x64 3.1.3.

EBLUPs based on a FH model

A basic area level model is the Fay-Herriot (FH) model, introduced by Fay and Herriot (1979) to obtain small area estimators of median income in U.S. small places. This model is defined in two stages. Let $\hat{\delta}_d^{DIR}$ be a direct estimator of δ_d. In the first stage, we assume that, given δ_d, $\hat{\delta}_d^{DIR}$ is an unbiased estimator of δ_d; more concretely,

$$\hat{\delta}_d^{DIR} = \delta_d + e_d, \quad e_d \overset{ind}{\sim} N(0, \psi_d), \quad d = 1, \ldots, D, \tag{1}$$

where ψ_d is the sampling variance of the direct estimator $\hat{\delta}_d^{DIR}$ given δ_d, assumed to be known for all $d = 1, \ldots, D$. In a second stage, we assume that the area parameters δ_d are linearly related with a p-vector $\mathbf{x}_d$ of area level auxiliary variables as follows,

$$\delta_d = \mathbf{x}_d^\top \boldsymbol{\beta} + u_d, \quad u_d \overset{ind}{\sim} N(0, A), \quad d = 1, \ldots, D. \tag{2}$$

Model (2) is called linking model because it relates all areas through the common regression coefficients $\boldsymbol{\beta}$, allowing us to borrow strength from all areas. Model (1) is called sampling model because it represents the uncertainty due to the fact that δ_d is unobservable and, instead of δ_d, we observe its direct estimator based on the sample, $\hat{\delta}_d^{DIR}$. Combining the two model components, we obtain the linear mixed model

$$\hat{\delta}_d^{DIR} = \mathbf{x}_d^\top \boldsymbol{\beta} + u_d + e_d, \quad e_d \overset{ind}{\sim} N(0, \psi_d), \quad d = 1, \ldots, D, \tag{3}$$

where

$$u_d \overset{ind}{\sim} N(0, A), \quad d = 1, \ldots, D,$$

and u_d is independent of e_d for all d. Normality is not needed for point estimation but it is required for the estimation of the mean squared error.

Henderson (1975) obtained the best linear unbiased predictor (BLUP) of a mixed effect, i.e., a linear combination of the fixed and random effects $\boldsymbol{\beta}$ and $\mathbf{u} = (u_1, \ldots, u_D)^T$. The BLUP of δ_d under FH model (3) is given by

$$\tilde{\delta}_d^{BLUP} = \mathbf{x}_d^\top \tilde{\boldsymbol{\beta}}(A) + \tilde{u}_d(A), \tag{4}$$

where $\tilde{u}_d(A) = \gamma_d(A)\left(\hat{\delta}_d^{DIR} - \mathbf{x}_d^\top \tilde{\boldsymbol{\beta}}(A)\right)$ is the predicted random effect, $\gamma_d(A) = A/(A + \psi_d) \in (0, 1)$ and $\tilde{\boldsymbol{\beta}}(A) = \left[\sum_{d=1}^{D} (A + \psi_d)^{-1} \mathbf{x}_d \mathbf{x}_d^T\right]^{-1} \sum_{d=1}^{D} (A + \psi_d)^{-1} \mathbf{x}_d \hat{\delta}_d^{DIR}$ is the weighted least squares estimator of $\boldsymbol{\beta}$.

The BLUP assumes that A is known. The empirical BLUP (EBLUP) $\hat{\delta}_d^{EBLUP}$ is obtained by replacing A in the BLUP (4) by a consistent estimator $\hat{A}$. The EBLUP can be expressed as a combination of the direct and the regression-synthetic estimators as follows,

$$\hat{\delta}_d^{EBLUP} = \hat{\gamma}_d \hat{\delta}_d^{DIR} + (1 - \hat{\gamma}_d)\mathbf{x}_d^\top \hat{\boldsymbol{\beta}}, \tag{5}$$

where $\hat{\gamma}_d = \gamma_d(\hat{A}) = \hat{A}/(\hat{A} + \psi_d)$ and $\hat{\boldsymbol{\beta}} = \tilde{\boldsymbol{\beta}}(\hat{A})$. In (5), we can see that when the direct estimator is reliable, i.e. ψ_d is small as compared with $\hat{A}$, then the EBLUP comes closer to the direct estimator. In contrast, when the direct estimator is unreliable, i.e. ψ_d is large as compared with $\hat{A}$, then the EBLUP gets closer to the regression-synthetic estimator. Thus, the EBLUP makes use of the regression assumption only for areas where borrowing strength is needed.

Common model fitting methods delivering consistent estimators for A are Fay-Herriot (FH) method (Fay and Herriot, 1979), maximum likelihood (ML) and restricted ML (REML), where the latter accounts for the degrees of freedom due to estimating $\boldsymbol{\beta}$ and therefore has a reduced finite sample bias. If the estimator $\hat{A}$ is an even and translation invariant function of the vector of direct estimates, which holds for FH, ML and REML fitting methods, then under symmetric distributions of random effects and errors, the EBLUP $\hat{\delta}_d^{EBLUP} = \tilde{\delta}_d^{BLUP}(\hat{A})$ remains unbiased (Kackar and Harville, 1984).

Models are typically compared based on goodness-of-fit measures such as the log-likelihood, the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). Under FH model (3), the log-likelihood is given by

$$\ell(A, \boldsymbol{\beta}) = -\frac{1}{2}\left[D\log(2\pi) + \sum_{d=1}^{D}\log(A + \psi_d) + \sum_{d=1}^{D}(A + \psi_d)^{-1}\left(\hat{\delta}_d^{DIR} - \mathbf{x}_d^\top \boldsymbol{\beta}\right)^2\right].$$

AIC and BIC are respectively obtained as

$$\text{AIC} = -2\ell(A, \boldsymbol{\beta}) + 2(p+1), \quad \text{BIC} = -2\ell(A, \boldsymbol{\beta}) + (p+1)\log(D).$$

Analogous formulas are applied in the remaining functions dealing with extensions of FH model, but using the corresponding log-likelihood. For functions based on BHF model, goodness-of-fit measures are those delivered by function `lme()` of the package **nlme**. A point estimate $\hat{\delta}_d$ of δ_d must be supplemented with an uncertainty measure; typically, the mean squared error $\text{MSE}(\hat{\delta}_d) = E(\hat{\delta}_d - \delta_d)^2$. The MSE of the EBLUP under the basic FH model (3) can be estimated analytically using the large sample approximation obtained by Prasad and Rao (1990) for a moments estimator of A. For REML and ML fitting methods, the analytical MSE estimates were firstly obtained by Datta and Lahiri (2000) and for FH fitting method, by Datta et al. (2005).

Functions `eblupFH()` and `mseFH()` calculate respectively small area estimates and corresponding analytical MSE estimates under FH model. The calls to these functions are

```
eblupFH(formula, vardir, method = "REML", MAXITER = 100, PRECISION = 0.0001, data)
mseFH(formula, vardir, method = "REML", MAXITER = 100, PRECISION = 0.0001, data)
```

Both functions require specification of the fixed part of FH model (3) through a usual R `formula` object, placing the vector of direct estimates on the left-hand side of `formula` and the desired area level covariates separated by "+" on the right-hand side. The `formula` automatically adds an intercept by default. These functions also require estimates of the sampling variances of the direct estimators in `vardir`. The direct estimates (left-hand side of `formula`) and their estimated variances (`vardir`) required in the area level functions can be previously obtained using the function `direct()` included in the package **sae** or using the R packages **survey** (Lumley, 2004, 2012) or **sampling** (Tillé and Matei, 2012) when the sampling design information is available. The default fitting method (`method`) is `REML` and it can be changed to `FH` and `ML`. Default maximum number of iterations (`MAXITER`) and convergence tolerance criteria (`PRECISION`) of the Fisher-scoring algorithm can be also modified. The last argument, `data`, can be used to specify a data object that contains the variables in `formula` and `vardir` as columns. The functions do not allow `NA` values because in area level models we do not consider areas with zero sample size.

The function `eblupFH()` returns a list with two objects: `eblup`, a vector with the EBLUPs for the areas, and `fit`, which includes all interesting output from the fitting process. The function `mseFH()` gives also the EBLUPs, but supplemented with their analytical MSE estimates. This function delivers a list with two objects: `est`, a list containing the EBLUPs and the results of the model fitting, and `mse`, a vector with the estimated MSEs.

Example 1. Average expenditure on fresh milk

We consider the data set `milk` on fresh milk expenditure, used originally by Arora and Lahiri (1997) and later by You and Chapman (2006). This data set contains 43 observations on the following six variables: SmallArea containing the areas of inferential interest, ni with the area sample sizes, yi with the average expenditure on fresh milk for the year 1989 (direct estimates), SD with the estimated standard deviations of direct estimators, CV with the estimated coefficients of variation of direct estimators and, finally, MajorArea containing major areas created by You and Chapman (2006). We will obtain EBLUPs $\hat{\delta}_d^{EBLUP}$ of average area expenditure on fresh milk for 1989, δ_d, together with analytical MSE estimates $\text{mse}(\hat{\delta}_d^{EBLUP})$, based on FH model with fixed effects for MajorArea categories. We will calculate the coefficients of variation (CVs) in terms of the MSE estimates as $\text{cv}(\hat{\delta}_d^{EBLUP}) = 100\sqrt{\text{mse}(\hat{\delta}_d^{EBLUP})}/\hat{\delta}_d^{EBLUP}$. We will analyze the gain in efficiency of the EBLUPs $\hat{\delta}_d^{EBLUP}$ in comparison with direct estimators $\hat{\delta}_d^{DIR}$ based on the CVs.

```
> data("milk")
> attach(milk)
> FH <- mseFH(yi ~ as.factor(MajorArea), SD^2)
> cv.FH <- 100 * sqrt(FH$mse) / FH$est$eblup
> results <- data.frame(Area = SmallArea, SampleSize = ni, DIR = yi,
+                       cv.DIR = 100 * CV, eblup.FH = FH$est$eblup, cv.FH)
> detach(milk)
```

EBLUPs and direct area estimates of average expenditure are plotted for each small area in Figure 1 left. CVs of these estimators are plotted in Figure 1 right. In both plots, small areas have been sorted by decreasing sample size. The following R commands are run to obtain Figures 1 left and right:

```
> results <- results[order(results$SampleSize, decreasing = TRUE), ]
> # Figure 1 left
> plot(results$DIR, type = "n", ylab = "Estimate", ylim = c(0.4, 1.6),
+      xlab = "area (sorted by decreasing sample size)", cex.axis = 1.5,
+      cex.lab = 1.5)
> points(results$DIR, type = "b", col = 1, lwd = 2, pch = 1, lty = 1)
> points(results$eblup.FH, type = "b", col = 4, lwd = 2, pch = 4, lty = 2)
> legend("top", legend = c("Direct", "EBLUP FH"), ncol = 2, col = c(1, 4), lwd = 2,
+        pch = c(1, 4), lty = c(1, 2), cex = 1.3)
> plot(results$cv.DIR, type = "n", ylab = "CV", ylim = c(5, 40),
+      xlab = "area (sorted by decreasing sample size)", cex.axis = 1.5,
+      cex.lab = 1.5)
> points(results$cv.DIR, type = "b", col = 1, lwd = 2, pch = 1, lty = 1)
> points(results$cv.FH, type = "b", col = 4, lwd = 2, pch = 4, lty = 2)
> legend("top", legend = c("Direct", "EBLUP FH"), ncol = 2, col = c(1, 4), lwd = 2,
+        pch = c(1, 4), lty = c(1, 2), cex = 1.3)
```

Observe in Figure 1 left that EBLUPs track direct estimators but are slightly less volatile. See also that CVs of EBLUPs are smaller than those of direct estimators for all areas in Figure 1 right. In fact, national statistical institutes are committed to publish statistical figures with a minimum level of reliability. A generally accepted rule is that an estimate with CV over 20% cannot be published. In this application, direct estimators have CVs over 20% for several areas, whereas the CVs of the EBLUPs do not exceed this limit for any of the areas. Moreover, the gains in efficiency of the EBLUPs tend to be larger for areas with smaller sample sizes (those on the right-hand side). Thus, in this example EBLUPs based on FH model seem more reliable than direct estimators.

EBLUPs based on a spatial FH model

In many practical applications, data from domains that are close to each other are correlated. One way to account for this correlation is by considering a spatial Fay-Herriot (SFH) model obtained by assuming that, in FH model given in (3), the vector $\mathbf{u} = (u_1, \ldots, u_D)^\top$ of domain effects follows a first order simultaneous autoregressive, SAR(1), process, that is,

$$\mathbf{u} = \rho_1 \mathbf{W}\mathbf{u} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}_D, \sigma_1^2 \mathbf{I}_D), \tag{6}$$

where $\mathbf{0}_k$ denotes a (column) vector of zeros of size k and $\mathbf{I}_k$ is the $k \times k$ identity matrix. In (6), $\rho_1 \in (-1, 1)$ is an unknown autorregression parameter and $\mathbf{W}$ is a $D \times D$ proximity matrix obtained by a row-wise standardization of an initial matrix with zeros on the diagonal and the remaining entries equal to one when the row domain is neighbor of the column domain, see e.g., Anselin (1988) and Cressie (1993).

The EBLUP under the SFH model (3) with area effects following (6) was obtained by Petrucci and Salvati (2006). The vector of EBLUPs for all areas are obtained with the function eblupSFH().

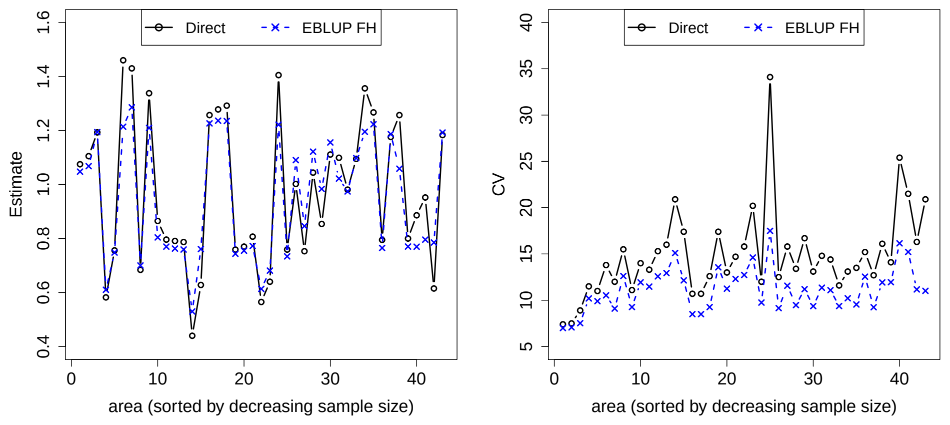


Figure 1: EBLUPs based on FH model and direct area estimates of average expenditure on fresh milk for each area (left). CVs of EBLUPs and direct estimators for each area (right). Areas are sorted by decreasing sample size.

Concerning MSE, Singh et al. (2005) gave an analytical estimator when model parameters are estimated either by ML or REML fitting methods. These analytical MSE estimates are implemented in function mseSFH(). Under complex models such as the SFH model, bootstrap methods are convenient alternatives because of their conceptual simplicity. Molina et al. (2009) provided parametric and non-parametric bootstrap procedures for estimation of the MSE under the SFH model. They can be obtained respectively with functions pbmseSFH() and npbmseSFH(). The calls to the functions related to the SFH model are:

```
eblupSFH(formula, vardir, proxmat, method = "REML", MAXITER = 100, PRECISION = 0.0001,
         data)
mseSFH(formula, vardir, proxmat, method = "REML", MAXITER = 100, PRECISION = 0.0001,
       data)
pbmseSFH(formula, vardir, proxmat, B = 100, method = "REML", MAXITER = 100,
         PRECISION = 0.0001, data)
npbmseSFH(formula, vardir, proxmat, B = 100, method = "REML", MAXITER = 100,
          PRECISION = 0.0001, data)
```

Some of the arguments are exactly the same as in the functions for FH model. The output has also the same structure. Additional arguments are a proximity matrix (proxmat), whose elements are proximities or neighborhoods of the areas, i.e., a matrix with elements in [0,1], zeros on the diagonal and rows adding up to 1. Functions using bootstrap methods also require to specify the number of bootstrap replicates B. In order to achieve stable MSE estimates, a large number of bootstrap replicates B is required. By default B is set to 100 to save computing time but we strongly recommend to set B to values over 200. Bootstrap functions are based on random number generation and the seed for random number generation can be fixed previously using set.seed(). The fitting method (method) can be chosen between REML (default value) or ML.

Example 2. Mean surface area for grape production

We consider now synthetic data on grape production for 274 municipalities in the region of Tuscany (Italy). The data set grapes contains the following variables: grapehect, direct estimators of the mean surface area (in hectares) used for production of grape for each municipality, area, agrarian surface area (in hectares) used for production, workdays, average number of working days in the reference year and var, sampling variance of the direct estimators for each municipality. We calculate spatial EBLUPs of mean surface area used for grape production, based on a spatial FH model with area and workdays as auxiliary variables, together with analytical MSE estimates. The data set grapesprox contains the proximity matrix representing the neighborhood structure of the municipalities in Tuscany.

We first load the two data sets, grapes and grapesprox. Then we call the function mseSFH() that returns small area estimates and analytical MSE estimates, calculate CVs and finally collect the obtained results in a data frame:

```
> data("grapes")
> data("grapesprox")
> SFH <- mseSFH(grapehect ~ area + workdays - 1, var, grapesprox, data = grapes)
```

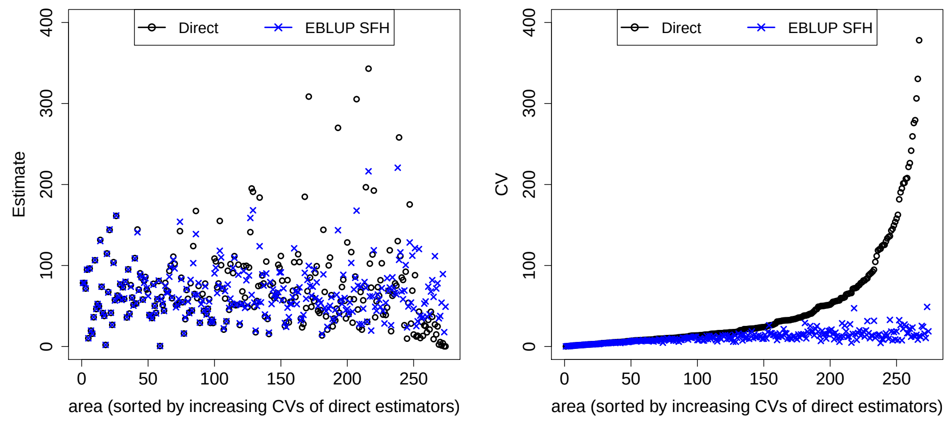


Figure 2: EBLUPs based on the SFH model and direct estimates of mean surface area used for production of grape for each municipality (left). CVs of EBLUPs and direct estimators for each municipality (right). Municipalities are sorted by increasing CVs of direct estimators.

```
> cv.SFH <- 100 * sqrt(SFH$mse) / SFH$est$eblup
> results <- data.frame(DIR = grapes$grapehect,
+                       cv.DIR = 100 * abs(sqrt(grapes$var) / grapes$grapehect),
+                       eblup.SFH = SFH$est$eblup, cv.SFH)
```

Figure 2 left shows the EBLUPs based on the SFH model and the direct estimates of mean surface area used for production of grape for each municipality. Figure 2 right shows the CVs of EBLUPs and direct estimators for each municipality, where municipalities are sorted by increasing CVs of direct estimators. Observe that the EBLUPs are again more stable and that CVs of EBLUPs are smaller for most municipalities, and the CV gains are remarkable for the municipalities in which direct estimators are very inefficient.

The following R commands are run to obtain Figures 2 left and right:

```
> # Sort results by increasing CV of direct estimators
> results <- results[order(results$cv.DIR), ]
> # Figure 2 left
> plot(results$DIR, type = "n", ylab = "Estimate", ylim = c(0, 400),
+      xlab = "area (sorted by increasing CVs of direct estimators)", cex.axis = 1.5,
+      cex.lab = 1.5)
> points(results$DIR, type = "p", col = 1, lwd = 2, pch = 1)
> points(results$eblup.SFH, type = "p", col = 4, lwd = 2, pch = 4)
> legend("top", legend = c("Direct", "EBLUP SFH"), ncol = 2, col = c(1, 4), lwd = 2,
+        pch = c(1, 4), cex = 1.3)
> # Figure 2 right
> plot(results$cv.DIR, type = "n", ylab = "CV", ylim = c(0, 400),
+      xlab = "area (sorted by increasing CVs of direct estimators)", cex.axis = 1.5,
+      cex.lab = 1.5)
> points(results$cv.DIR, type = "p", col = 1, lwd = 2, pch = 1)
> points(results$cv.SFH, type = "p", col = 4, lwd = 2, pch = 4)
> legend("top", legend = c("Direct", "EBLUP SFH"), ncol = 2, col = c(1, 4), lwd = 2,
+        pch = c(1, 4), cex = 1.3)
```

EBLUPs based on a spatio-temporal FH model

If area level data are available for several periods of time and for each area, the SFH model can be further extended to make use of this additional information. Consider that data are available for T time periods within each domain. Let δ_{dt} be the target parameter in domain d at time period t and let $\hat{\delta}_{dt}^{DIR}$ be a direct estimate of δ_{dt}. The STFH model proposed by Marhuenda et al. (2013) extends the SFH model by including random effects for the time periods nested within domains as follows

$$\hat{\delta}_{dt}^{DIR} = \mathbf{x}_d^\top \boldsymbol{\beta} + u_d + v_{dt} + e_{dt}, \quad e_{dt} \overset{ind}{\sim} N(0, \psi_d), \quad t = 1, \ldots, T, \quad d = 1, \ldots, D.$$

Here, the vector $\mathbf{u} = (u_1, \ldots, u_D)^\top$ of area effects follows the SAR(1) process given in (6) and, for each area d, the vectors $\mathbf{v}_d = (v_{d1}, \ldots, v_{dT})^\top$ are i.i.d. following the first order autoregressive, AR(1), process

$$v_{dt} = \rho_2 v_{d,t-1} + \epsilon_{2dt}, \quad \epsilon_{2dt} \sim N(0, \sigma_2^2).$$

Much more complex models than the AR(1) process are not typically considered in small area estimation because in practical applications the number of available time periods T is typically small. Moving average (MA) processes are not yet considered in the **sae** package.

Marhuenda et al. (2013) give the EBLUP of $\hat{\delta}_{dt}$ under the STFH model and provide a parametric bootstrap procedure for the estimation of the MSE of the EBLUP. EBLUPs for all areas and parametric bootstrap estimates can be obtained calling functions `eblupSTFH()` and `pbmseSTFH()` respectively. The calls to these functions are:

```
eblupSTFH(formula, D, T, vardir, proxmat, model = "ST", MAXITER = 100,
          PRECISION = 0.0001, data)
pbmseSTFH(formula, D, T, vardir, proxmat, B = 100, model = "ST", MAXITER = 100,
          PRECISION = 0.0001, data)
```

The arguments of these functions are the same as for the SFH model with the exception that argument `method` is not used because currently only the REML fitting method has been implemented for the STFH functions. Additionally, we must specify the number of areas `D` and the number of periods of time `T` for each area. Note that these functions can be used only when data are available for all the `T` periods of time within each of the `D` domains. Data in `formula` and `vardir` must be sorted in ascending order by period of time for each domain. The argument `model` can be chosen between the default value `ST` (AR(1) time-effects within each domain) or value `S` (uncorrelated time effects within each domain). For the bootstrap method, again we recommend to take at least `B`=200 bootstrap replicates. Again, the output of these functions has the same structure as that of functions for FH model.

Example 3. EBLUPs based on a spatio-temporal FH model

In this example, we use the data set `spacetime`, which contains synthetic area level data for $T = 3$ periods of time for each of $D = 11$ areas. The data set contains the following variables: `Area`, area code, `Time`, period of time, `X1` and `X2`, the auxiliary variables for each area and period of time, `Y`, direct estimates for each area and period of time and `Var`, sampling variances of the direct estimators. We calculate EBLUPs of the means for each area and period of time, based on the STFH model with proximity matrix given in the data set `spacetimeprox`, together with parametric bootstrap MSE estimates. We show the results only for the last period of time.

```
> data("spacetime")
> data("spacetimeprox")
> D <- nrow(spacetimeprox)              # number of areas
> T <- length(unique(spacetime$Time))   # number of time periods
> set.seed(123)
> STFH <- pbmseSTFH(Y ~ X1 + X2, D, T, vardir = Var, spacetimeprox, data = spacetime)

Bootstrap procedure with B = 100 iterations starts.
b = 1
...
b = 100

> # Compute CVs for the EBLUPs based on the STFH model and for the direct estimators
> cv.STFH <- 100 * sqrt(STFH$mse) / STFH$est$eblup
> cv.DIR <- 100 * sqrt(spacetime$Var) / spacetime$Y
> results <- data.frame(Area = spacetime$Area, Time = spacetime$Time,
+                       DIR = spacetime$Y, eblup.STFH = STFH$est$eblup,
+                       cv.DIR, cv.STFH)
> results.lasttime <- results[results$Time == 3, ]
> print(results.lasttime, row.names = FALSE)

 Area Time      DIR eblup.STFH    cv.DIR   cv.STFH
    2    3 0.261484 0.27343181 10.944523  7.653997
    3    3 0.175358 0.17722992  7.777336  7.026746
    8    3 0.096230 0.09653879  6.059391  5.567674
   12    3 0.122160 0.13740348 21.904205 14.798918
```

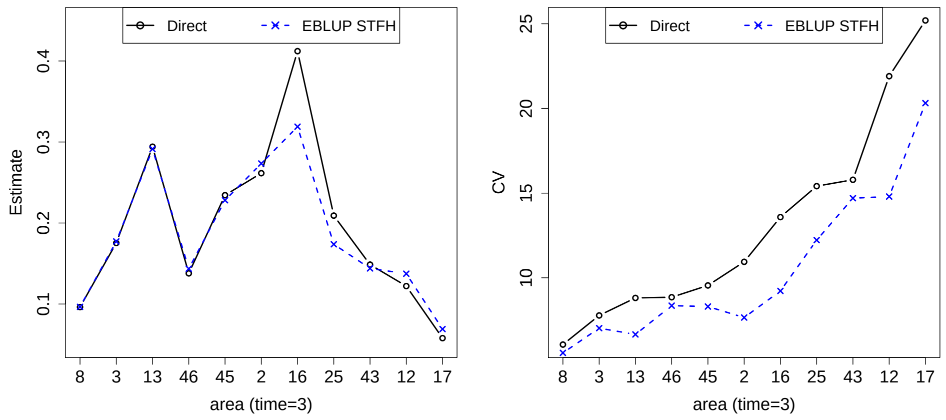


Figure 3: EBLUPs of each area mean at last period of time, based on the STFH model and direct estimates (left). CVs of the two estimators for each area (right). Areas are sorted by increasing CVs of direct estimators.

```
13     3 0.294176 0.29129477  8.812059  6.657347
16     3 0.412106 0.31887378 13.584403  9.224897
17     3 0.057924 0.06912566 25.195980 20.314774
25     3 0.209146 0.17377084 15.411972 12.225196
43     3 0.148671 0.14398844 15.788815 14.700855
45     3 0.234361 0.22810227  9.550663  8.303303
46     3 0.137869 0.14354272  8.853735  8.355827
```

Figure 3 left shows the EBLUPs based on the STFH model together with the direct estimates for each area at the last time point, with areas sorted by increasing CVs of direct estimators. Figure 3 right shows the corresponding CVs. In this example, we can see that even with a very small number of areas $D = 11$ and periods of time $T = 3$ to borrow strength from, the EBLUPs follow closely direct estimates but are still slightly more stable and the CVs of EBLUPs are smaller for all areas.

The following R commands are executed to obtain Figures 3 left and right:

```
> results.lasttime <- results.lasttime[order(results.lasttime$cv.DIR), ]
> # Figure 3 left
> plot(results.lasttime$DIR, type = "n", xlab = "area (time=3)", ylab = "Estimate",
+      ylim = c(0.05, 0.45), cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
> axis(1, 1:11, results.lasttime$Area, cex.axis = 1.5)
> points(results.lasttime$DIR, type = "b", col = 1, lwd = 2, pch = 1, lty = 1)
> points(results.lasttime$eblup.STFH, type = "b", col = 4, lwd = 2, pch = 4, lty = 2)
> legend("top", legend = c("Direct", "EBLUP STFH"), ncol = 2, col = c(1, 4), lwd = 2,
+        pch = c(1, 4), lty = c(1, 2), cex = 1.3)
> # Figure 3 right
> plot(results.lasttime$cv.DIR, type = "n", xlab = "area (time=3)", ylab = "CV",
+      cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
> axis(1, 1:11, results.lasttime$Area, cex.axis = 1.5)
> points(results.lasttime$cv.DIR, type = "b", col = 1, lwd = 2, pch = 1, lty = 1)
> points(results.lasttime$cv.STFH, type = "b", col = 4, lwd = 2, pch = 4, lty = 2)
> legend("top", legend = c("Direct", "EBLUP STFH"), ncol = 2, col = c(1, 4), lwd = 2,
+        pch = c(1, 4), lty = c(1, 2), cex = 1.3)
```

EBLUPs based on BHF model

When auxiliary data are available at the unit level, unit-level models are likely to provide more efficient small area estimators than area level models, because they make use of the much richer information offered by microdata. Let the data for unit j in area d be $(Y_{dj}, \mathbf{x}_{dj}^\top)$, where $\mathbf{x}_{dj}$ is the vector with the values of p auxiliary variables. The basic unit-level model is the nested error model introduced by Battese et al. (1988), given by

$$Y_{dj} = \mathbf{x}_{dj}^\top \boldsymbol{\beta} + u_d + e_{dj}, \quad u_d \overset{iid}{\sim} N(0, \sigma_u^2), \quad e_{dj} \overset{iid}{\sim} N(0, \sigma_e^2). \tag{7}$$

Here, u_d are area effects and e_{dj} are individual errors, where u_d and e_{dj} are assumed to be independent with corresponding variances σ_u^2 and σ_e^2, regarded as unknown parameters. The model defined in (7) is assumed for all units in the population and we consider that sample selection bias is absent and therefore sample units follow exactly the same model.

For the estimation of a linear parameter $\delta_d = \mathbf{a}_d^\top \mathbf{y}_d$ under BHF model, Royall (1970) derived the BLUP. As a particular case, for the small area mean $\delta_d = \bar{Y}_d = N_d^{-1} \sum_{j=1}^{N_d} Y_{dj}$, the BLUP is given by

$$\tilde{\bar{Y}}_d^{BLUP} = \frac{1}{N_d}\left(\sum_{j\in s_d} Y_{dj} + \sum_{j\in r_d} \tilde{Y}_{dj}\right),$$

where $\tilde{Y}_{dj} = \mathbf{x}_{dj}^\top \tilde{\boldsymbol{\beta}} + \tilde{u}_d$ is the BLUP of Y_{dj}. Here, $\tilde{u}_d$ is the BLUP of u_d, given by $\tilde{u}_d = \gamma_d(\bar{y}_{ds} - \bar{\mathbf{x}}_{ds}^\top \tilde{\boldsymbol{\beta}})$, with $\bar{y}_{ds} = n_d^{-1}\sum_{j\in s_d} Y_{dj}$, $\bar{\mathbf{x}}_{ds} = n_d^{-1}\sum_{j\in s_d} \mathbf{x}_{dj}$ and $\gamma_d = \sigma_u^2/(\sigma_u^2 + \sigma_e^2/n_d)$. Again, the BLUP of $\bar{Y}_d$ depends on the vector $\boldsymbol{\theta} = (\sigma_u^2, \sigma_e^2)^\top$ of unknown variance components. In practice, we calculate the EBLUP of $\bar{Y}_d$ by replacing $\boldsymbol{\theta} = (\sigma_u^2, \sigma_e^2)^\top$ by a consistent estimator $\hat{\boldsymbol{\theta}} = (\hat{\sigma}_u^2, \hat{\sigma}_e^2)^\top$. Usual fitting methods for BHF model are also ML and REML. Let $\hat{\boldsymbol{\beta}}$ and $\hat{u}_d = \hat{\gamma}_d(\bar{y}_{ds} - \bar{\mathbf{x}}_{ds}^\top \hat{\boldsymbol{\beta}})$ be the results of replacing $\boldsymbol{\theta}$ by $\hat{\boldsymbol{\theta}}$ in the formulas of $\tilde{\boldsymbol{\beta}}$ and $\tilde{u}_d$, where $\hat{\gamma}_d = \hat{\sigma}_u^2/(\hat{\sigma}_u^2 + \hat{\sigma}_e^2/n_d)$. Then the EBLUP of $\bar{Y}_d$ can be alternatively expressed as

$$\hat{\bar{Y}}_d^{EBLUP} = f_d\,\bar{y}_{ds} + (\bar{\mathbf{X}}_d - f_d\,\bar{\mathbf{x}}_{ds})^\top\,\hat{\boldsymbol{\beta}} + (1 - f_d)\hat{u}_d, \tag{8}$$

where $f_d = n_d/N_d$ is the domain sampling fraction. Equation (8) shows that, for calculation of the EBLUP of a small area mean, apart from sample observations, we need the true totals or means $\bar{\mathbf{X}}_d$ of the auxiliary variables in the population and the populations sizes N_d of the areas.

For MSE estimation of the EBLUP given in (8) based on BHF model, González-Manteiga et al. (2008) proposed a parametric bootstrap method for finite populations. EBLUPs of the area means based on BHF model given in (7) and parametric bootstrap MSE estimates can be obtained from the functions `eblupBHF()` and `pbmseBHF()` respectively. The calls to these functions are:

```
eblupBHF(formula, dom, selectdom, meanxpop, popnsize, method = "REML", data)
pbmseBHF(formula, dom, selectdom, meanxpop, popnsize, B = 200,  method = "REML", data)
```

The fixed part of the model needs to be specified through the argument `formula` and the variable (vector or factor) identifying the domains must be specified in the argument `dom`. The variables in `formula` and `dom` can also be chosen from a data set specified in the argument `data`. These two functions allow selection of a subset of domains where we want to estimate by specifying the vector of selected domains in `selectdom`, which by default includes the list of all unique domains in `dom`. The population means of the auxiliary variables for the domains (`meanxpop`) and the population sizes of the domains (`popnsize`) are required arguments. `REML` (default) or `ML` fitting methods can be specified in argument `method`. The output of these functions has the same structure as that of FH functions. In these functions, the observations with `NA` values in `formula` or `dom` are ignored. These functions deliver estimates for areas with zero sample size, that is, for areas specified in `selectdom` without observations in `formula`, as long as these areas have elements in `meanxpop`. In this case, the function delivers the synthetic estimator $\hat{\bar{Y}}_d^{EBLUP} = \bar{\mathbf{X}}_d^\top \hat{\boldsymbol{\beta}}$.

Example 4. County means of corn crop hectares

We consider data used in Battese et al. (1988) on corn and soy beans production in 12 Iowa counties, contained in the two data sets `cornsoybean` and `cornsoybeanmeans`. Data come from two different sources: the 1978 June Enumerative Survey of the U.S. Department of Agriculture and images of land observatory satellites (LANDSAT) during the 1978 growing season.

In these data sets, counties are the domains and sample segments are the units. The data set `cornsoybean` contains the values of the following variables for each sample segment within each county: `County`, county code, `CornHec`, reported hectares of corn from the survey in each sample segment within each county, `SoyBeansHec`, reported hectares of soy beans from the survey in each sample segment within county, `CornPix`, number of pixels of corn from satellite data, and `SoyBeansPix`, number of pixels of soy beans from satellite data.

In this example, we will calculate EBLUPs of county means of corn crop hectares based on BHF model, considering as auxiliary variables the number of pixels of corn and soy beans from the LANDSAT satellite images. See from (8) that the domain (county) means of the auxiliary variables $\bar{\mathbf{X}}_d$ and the population sizes N_d of the counties are required to obtain the EBLUPs based on BHF model. These county means are included in the data set `cornsoybeanmeans`. Concretely, this data set contains: `SampSegments`, number of sample segments in the county (sample size), `PopnSegments`, number of population segments in the county (population size), `MeanCornPixPerSeg`, county mean of the number of corn pixels per segment, and `MeanSoyBeansPixPerSeg`, county mean of the number of soy beans pixels per segment (county means of auxiliary variables).

First, we create the data frame `Xmean` containing the true county means of the auxiliary variables given in the columns named `MeanCornPixPerSeg` and `MeanSoyBeansPixPerSeg` from the data set `cornsoybeanmeans`. We also create the data frame `Popn` containing the county population sizes. In these two data frames, the first column must contain the domain (or county) codes. Although here counties in `Xmean` and `Popn` are sorted exactly in the same way, the functions for BHF model handle correctly the case in which the domains (whose codes are listed in the first column of both `Xmean` and `Popn`) are arranged differently:

```
> data("cornsoybeanmeans")
> Xmean <- data.frame(cornsoybeanmeans[, c("CountyIndex", "MeanCornPixPerSeg",
+                                          "MeanSoyBeansPixPerSeg")])
> Popn <- data.frame(cornsoybeanmeans[, c("CountyIndex", "PopnSegments")])
```

Next, we load the data set with the unit-level data and delete observation number 33 because it is an outlier, see Battese et al. (1988). Then we call the function `pbmseBHF()`, which gives the EBLUPs of the means of corn crop area and parametric bootstrap MSE estimates, choosing `B=200` bootstrap replicates. Here, `CornHec` is the response variable and the auxiliary variables are `CornPix` and `SoyBeansPix`.

Note that the argument `selectdom` can be used to select a subset of the domains for estimation.

```
> data("cornsoybean")
> cornsoybean <- cornsoybean[-33, ]
> set.seed(123)
> BHF <- pbmseBHF(CornHec ~ CornPix + SoyBeansPix, dom = County, meanxpop = Xmean,
+                 popnsize = Popn, B = 200, data = cornsoybean)
Bootstrap procedure with B = 200 iterations starts.
b = 1
...
b = 200
```

Finally, we compute CVs and construct a data frame with sample sizes, EBLUPs and CVs for each county, called `results.corn`.

```
> cv.BHF <- 100 * sqrt(BHF$mse$mse) / BHF$est$eblup$eblup
> results <- data.frame(CountyIndex = BHF$est$eblup$domain,
+                       CountyName = cornsoybeanmeans$CountyName,
+                       SampleSize = BHF$est$eblup$sampsize,
+                       eblup.BHF = BHF$est$eblup$eblup, cv.BHF)
> print(results, row.names = FALSE)
 CountyIndex CountyName SampleSize eblup.BHF   cv.BHF
           1 CerroGordo          1  122.1954 8.066110
           2   Hamilton          1  126.2280 7.825271
           3      Worth          1  106.6638 9.333344
           4   Humboldt          2  108.4222 7.598736
           5   Franklin          3  144.3072 4.875002
           6 Pocahontas          3  112.1586 6.020232
           7  Winnebago          3  112.7801 5.951520
           8     Wright          3  122.0020 5.700670
           9    Webster          4  115.3438 4.808813
          10    Hancock          5  124.4144 4.495448
          11    Kossuth          5  106.8883 4.532518
          12     Hardin          5  143.0312 3.504340
```

Results show great similarity with those given in Battese et al. (1988) although the model fitting method and the MSE estimation procedure used here are different.

EB estimators of nonlinear parameters based on BHF model

Now consider that we wish to estimate a general nonlinear area parameter $\delta_d = h(\mathbf{y}_d)$, where $\mathbf{y}_d = (Y_{d1}, \ldots, Y_{dN_d})^\top$ is the vector of measurements of the response variable in the units from area d. Rearranging the elements Y_{dj} according to their membership to the sample s_d or the sample complement r_d, we can express $\mathbf{y}_d$ as $\mathbf{y}_d = (\mathbf{y}_{ds}^\top, \mathbf{y}_{dr}^\top)^\top$, where $\mathbf{y}_{ds}$ and $\mathbf{y}_{dr}$ denote respectively the subvectors containing the sample and out-of-sample elements. When $\delta_d = h(\mathbf{y}_d)$ is nonlinear in $\mathbf{y}_d$, considering a linear predictor like the BLUP makes no sense; instead, we consider the best predictor, which minimizes the MSE without restrictions of linearity or unbiasedness. The best predictor is given by

$$\tilde{\delta}_d^B = E_{\mathbf{y}_{dr}}\left[h(\mathbf{y}_d)|\mathbf{y}_{ds}\right] = \int h(\mathbf{y}_d) f(\mathbf{y}_{dr}|\mathbf{y}_{ds}) d\mathbf{y}_{dr}, \tag{9}$$

where the expectation is taken with respect to the distribution of $\mathbf{y}_{dr}$ given $\mathbf{y}_{ds}$, with density $f(\mathbf{y}_{dr}|\mathbf{y}_{ds})$. Under BHF model for Y_{dj} given in (7), the distribution of $\mathbf{y}_{dr}$ given $\mathbf{y}_{ds}$ is normal with conditional mean vector and covariance matrix depending on the unknown parameters $\boldsymbol{\beta}$ and $\boldsymbol{\theta} = (\sigma_u^2, \sigma_e^2)^\top$. Even if this conditional distribution was completely known, the expected value in (9) would be still intractable for complex nonlinear parameters $\delta_d = h(\mathbf{y}_d)$ like some poverty indicators. For such cases, Molina and Rao (2010) propose to estimate the unknown model parameters by consistent estimators $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\theta}} = (\hat{\sigma}_u^2, \hat{\sigma}_e^2)^\top$ such as ML or REML estimators, and then obtaining the empirical best (EB) estimator of δ_d by a Monte Carlo approximation of the expected value in (9). This process is done by first generating out-of-sample vectors $\mathbf{y}_{dr}^{(\ell)}$, $\ell = 1, \dots, L$, for large L, from the (estimated) conditional distribution $f(\mathbf{y}_{dr}|\mathbf{y}_{ds}; \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}})$. The second step consists of attaching, for each ℓ, the sample elements to the generated vector $\mathbf{y}_{dr}^{(\ell)}$, resulting in the full population vector (or census) $\mathbf{y}_d^{(\ell)} = (\mathbf{y}_{ds}^\top, (\mathbf{y}_{dr}^{(\ell)})^\top)^\top$. With the census $\mathbf{y}_d^{(\ell)}$, we then calculate the target quantity $h(\mathbf{y}_d^{(\ell)})$ for each $\ell = 1, \dots, L$. Lastly, we average the target quantity over the L simulations as

$$\hat{\delta}_d^{EB} \approx \frac{1}{L} \sum_{\ell=1}^{L} h(\mathbf{y}_d^{(\ell)}). \tag{10}$$

Note that the size of $\mathbf{y}_{dr}$ is $N_d - n_d$, where N_d is typically large and n_d is typically small. Then, generation of $\mathbf{y}_{dr}$ might be computationally cumbersome. However, the generation of large multivariate normal vectors can be avoided by exploiting the form of the conditional covariance obtained from model (7). It is easy to see that out-of-sample vectors $\mathbf{y}_{dr}$ from the desired conditional distribution $f(\mathbf{y}_{dr}|\mathbf{y}_{ds}; \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}})$ can be obtained by generating only univariate variables from the following model

$$\begin{aligned} Y_{dj}^{(\ell)} &= \mathbf{x}_{dj}^\top \hat{\boldsymbol{\beta}} + \hat{u}_d + v_d + \varepsilon_{dj}, \\ v_d &\sim N(0, \hat{\sigma}_u^2(1 - \hat{\gamma}_d)),\ \varepsilon_{dj} \sim N(0, \hat{\sigma}_e^2),\ j \in r_d,\ d = 1, \dots, D. \end{aligned} \tag{11}$$

In some cases, the response variable in BHF model is a one-to-one transformation of the variable of interest, that is, $Y_{dj} = T(O_{dj})$, where O_{dj} are the measurements of the variable of interest in the population units. This situation often occurs in socio-economic applications. A good example is when O_{dj} is a variable measuring welfare of individuals such as income, and the target parameter for each area is a poverty indicator such as the poverty incidence, also called at-risk-of-poverty rate. The poverty incidence is defined as the proportion of people with income O_{dj} below the poverty line z, that is,

$$\delta_d = \frac{1}{N_d} \sum_{j=1}^{N_d} I\left(O_{dj} < z\right), \quad d = 1, \dots, D. \tag{12}$$

The distribution of incomes O_{dj} is typically severely skewed and therefore assuming BHF model for O_{dj} with normally distributed random effects and errors is not realistic. Thus, we cannot obtain EB estimates of δ_d based on BHF model for O_{dj} as described above. However, a transformation $Y_{dj} = T(O_{dj})$, such as $Y_{dj} = \log(O_{dj} + c)$ often leads to (at least approximate) normality. Observe that the target area parameter δ_d, if initially defined in terms of the original variables O_{dj}, can be easily expressed in terms of the transformed variables Y_{dj} using the inverse transformation $O_{dj} = T^{-1}(Y_{dj})$, as

$$\delta_d = h(\mathbf{y}_d) = \frac{1}{N_d} \sum_{j=1}^{N_d} I\left(T^{-1}(Y_{dj}) < z\right), \quad d = 1, \dots, D.$$

Expressing the target area parameter δ_d in terms of the actual model responses Y_{dj} for area d as $\delta_d = h(\mathbf{y}_d)$, we can then compute the Monte Carlo approximation of the EB estimate $\hat{\delta}_d^{EB}$ of δ_d as indicated in (10).

Suitable transformations $T()$ leading to normality can be found within the Box-Cox or power families. For a constant c and a power λ, the Box-Cox transformation is given by

$$T_{c,\lambda}(O_{dj}) = \begin{cases} \left[(O_{dj} + c)^\lambda - 1\right] / \lambda, & \lambda \neq 0; \\ \log(O_{dj} + c), & \lambda = 0. \end{cases}$$

The power family is given by

$$T^*_{c,\lambda}(O_{dj}) = \begin{cases} (O_{dj}+c)^\lambda, & \lambda \neq 0; \\ \log(O_{dj}+c), & \lambda = 0. \end{cases}$$

The log transformation is obtained in the two families setting $\lambda = 0$. MSE estimates of the EB estimators of $\delta_d = h(\mathbf{y}_d)$ under BHF model can be obtained using the parametric bootstrap method for finite populations introduced by González-Manteiga et al. (2008).

Function ebBHF() gives EB estimates of the area parameters $\delta_d = h(\mathbf{y}_d)$, where $Y_{dj} = T(O_{dj})$, based on BHF model for Y_{dj}. Function pbmseebBHF() gives EB estimates together with parametric bootstrap MSE estimates. The calls to these functions are:

```
ebBHF(formula, dom, selectdom, Xnonsample, MC = 100, data, transform = "BoxCox",
      lambda = 0, constant = 0, indicator)
pbmseebBHF(formula, dom, selectdom, Xnonsample, B = 100, MC = 100, data,
           transform = "BoxCox", lambda = 0, constant = 0, indicator)
```

In the left-hand side of the formula object, we must put the vector of observations of the original variables O_{dj}. We can either select "Box-Cox" or "power" family of transformations for O_{dj} through the argument transform. The constant c of the transformations is specified through the argument constant and the power is specified in lambda. This parameter is by default set to 0, which corresponds to log transformation for both families. Setting lambda equal to 1 and letting the default constant equal to 0 implies no transformation. Note that these functions assumes BHF model for the transformed variables $Y_{dj} = T(O_{dj})$. In the argument indicator, we must provide the name of the R function that is applied to the untransformed variables O_{dj} to obtain the target area parameter. For example, if the target parameters are the area medians of the original untransformed variables O_{dj}, we must specify indicator=median regardless of the transformation taken.

The two above functions require the values of auxiliary variables for each out-of-sample unit (Xnonsample). Additionally, in argument MC we must specify the number of Monte Carlo samples L that we wish to use in the Monte Carlo approximation of the EB estimators given in Equation (10). These functions ignore the observations with NA values in formula or dom. They also deliver EB estimates for areas with zero sample size, that is, for the areas specified in selectdom without any observations in formula, as long as the values of the auxiliary variables for all the units in these areas are included in Xnonsample. In this case, the functions generate the out-of-sample elements $Y^{(\ell)}_{dj}$ from model (11) with $\hat{\gamma}_d = 0$.

The function ebBHF() delivers a list of two objects: the first one is called eb and contains the EB estimates for each selected domain, and the second one is called fit, and contains a summary of the model fitting. The function pbmseebBHF() obtains the parametric bootstrap MSE estimates together with the EB estimates. It delivers a list with two objects. The first one, est, is another list containing itself two objects: the results of the point estimation process (eb) and a summary of the fitting process (fit). The second one, mse, contains a data frame with the estimated MSEs for each selected domain.

Example 5. Poverty mapping

In this example, we will illustrate how to estimate poverty incidences in Spanish provinces (areas). As given in Equation (12), the poverty incidence for a province is the province mean of a binary variable O_{dj} taking value 1 when the person's income is below the poverty line z and 0 otherwise.

The data set incomedata contains synthetic unit-level data on income and other sociological variables in the Spanish provinces. These data have been obtained by simulation, with the only purpose of being able to illustrate the use of the package functions. Therefore, conclusions regarding the levels of poverty in the Spanish provinces obtained from these data are not realistic. We will use the following variables from the data set: province name (provlab), province code (prov), income (income), sampling weight (weight), education level (educ), labor status (labor), and finally the indicators of each of the categories of educ and labor.

We will obtain EB estimates of province poverty incidences based on BHF model for the variable income. Note that the EB method assumes that the response variable considered in BHF model is (approximately) normally distributed. However, the histogram of income appears to be highly right-skewed and therefore transformation to achieve approximate normality is necessary. We select the log transformation, which is a member of both Box-Cox and power family, taking $\lambda = 0$. For the constant c in these transformation families, we tried with a grid of values in the range of income. For each value of c in the grid, we fitted BHF model to $\log(\text{income} + c)$ and selected the value of c for which the distribution of residuals was approximately symmetric, see the residual plots in Figure 4. The resulting value of the constant was $c = 3500$.

The package functions dealing with EB method can estimate whatever domain target parameter as desired, provided this target parameter is a function of a continuous variable O_{dj} whose transformation Y_{dj} will act as response variable in BHF model. We just need to define the target parameter for the domains as an R function. The target parameter in this example is the poverty incidence, which is a function of the continuous variable income. Thus, we define the R function povertyincidence() as function of y=income, considering as poverty line z = 6557.143.

```
> povertyincidence <- function(y) {
+   result <- mean(y < 6557.143)
+   return (result)
+ }
```

When estimating nonlinear parameters, the values of the auxiliary variables in the model are needed for each out-of-sample unit. Although we will use the sample data from all the provinces to fit the model, to save computation time here we will compute EB estimates and corresponding MSE estimates only for the 5 provinces with the smallest sample sizes. For these selected provinces, the data set Xoutsamp contains the values for each out-of-sample individual of the considered auxiliary variables, which are the categories of education level and of labor status, defined exactly as in the data set incomedata. Again, these data have been obtained by simulation.

We read the required data sets, create the vector provincecodes with province codes for the selected provinces and create also Xoutsamp_AuxVar, containing the values of the auxiliary variables for all out-of-sample individuals in these provinces.

```
> data("incomedata")
> data("Xoutsamp")
> provincecodes <- unique(Xoutsamp$domain)
> provincelabels <- unique(incomedata$provlab)[provincecodes]
> Xoutsamp_AuxVar <- Xoutsamp[ ,c("domain", "educ1", "educ3", "labor1", "labor2")]
```

Next, we use the function ebBHF to calculate EB estimates of the poverty incidences under BHF model for log(income+constant) for the 5 selected provinces specified in the argument selectdom. In the argument indicator, we must specify the function povertyincidence() defining the target parameter.

```
> set.seed(123)
> EB <- ebBHF(income ~ educ1 + educ3 + labor1 + labor2, dom = prov,
+             selectdom = provincecodes, Xnonsample = Xoutsamp_AuxVar, MC = 50,
+             constant = 3500, indicator = povertyincidence, data = incomedata)
```

The list fit of the output gives information about the fitting process. For example, we can see whether auxiliary variables are significant.

```
> EB$fit$summary
Linear mixed-effects model fit by REML
 Data: NULL
       AIC      BIC    logLik
  18980.72 19034.99 -9483.361

Random effects:
 Formula: ~1 | as.factor(dom)
        (Intercept)  Residual
StdDev:  0.09436138 0.4179426

Fixed effects: ys ~ -1 + Xs
                  Value   Std.Error    DF  t-value p-value
Xs(Intercept)  9.505176 0.014384770 17143 660.7805       0
Xseduc1       -0.124043 0.007281270 17143 -17.0359       0
Xseduc3        0.291927 0.010366323 17143  28.1611       0
Xslabor1       0.145985 0.006915979 17143  21.1084       0
Xslabor2      -0.081624 0.017082634 17143  -4.7782       0
 Correlation:
         Xs(In) Xsedc1 Xsedc3 Xslbr1
Xseduc1  -0.212
Xseduc3  -0.070  0.206
Xslabor1 -0.199  0.128 -0.228
Xslabor2 -0.079  0.039 -0.039  0.168
```

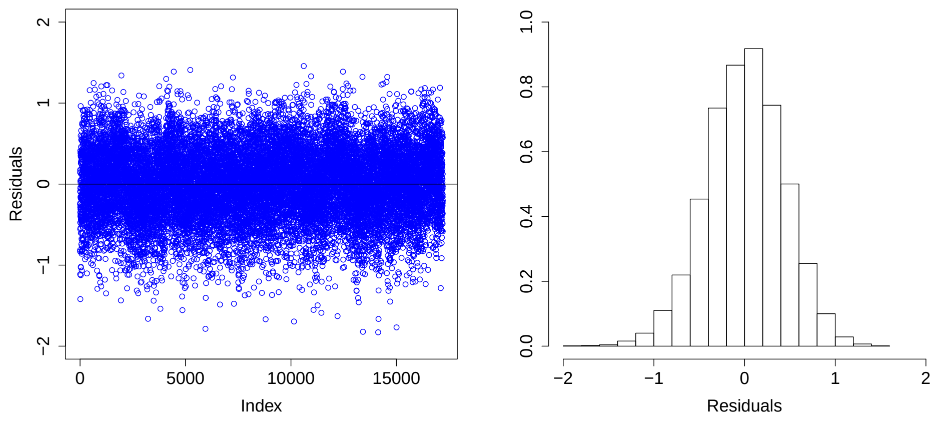


Figure 4: Index plot of residuals (left) and histogram of residuals (right) from the fitting of BHF model to log(income+constant).

```
Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-4.2201202 -0.6617181  0.0203607  0.6881828  3.5797393

Number of Observations: 17199
Number of Groups: 52
```

Checking model assumptions is crucial since the optimality properties of the EB estimates depend on the extent to which those assumptions are true. To detect departures from BHF model for the transformed income, we can draw the usual residual plots. The following commands draw an index plot of residuals and a histogram:

```
# Figure 4 left
> plot(EB$fit$residuals, xlab = "Index", ylab = "Residuals", cex.axis = 1.5,
+      cex.lab = 1.5, ylim = c(-2, 2), col = 4)
> abline(h = 0)

> # Figure 4 right
> hist(EB$fit$residuals, prob = TRUE, xlab = "Residuals", ylab = "", main = "",
+      cex.axis = 1.5, cex.lab = 1.5, xlim = c(-2, 2), ylim = c(0, 1))
```

These two plots appear in Figure 4, which shows no evidence of serious model departure.

Finally, we compute parametric bootstrap MSE estimates and calculate CVs of EB estimates. This process might be slow for a large number of bootstrap or Monte Carlo replicates `B` and `MC` respectively, large sample size or large number of auxiliary variables. Function `pbmseebBHF()` gives also EB estimates apart from MSEs:

```
> set.seed(123)
> pbmse.EB <- pbmseebBHF(income ~ educ1 + educ3 + labor1 + labor2, dom = prov,
+                        selectdom = provincecodes, Xnonsample = Xoutsamp_AuxVar,
+                        B = 200, MC = 50, constant = 3500,
+                        indicator = povertyincidence, data = incomedata)
Bootstrap procedure with B = 200 iterations starts.
b = 1
...
b = 200
> pbcv.EB <- 100 * sqrt(pbmse.EB$mse$mse) / abs(pbmse.EB$est$eb$eb)    # compute CV
```

Finally, in the data frame `results.EB` we collect the sample sizes, EB estimates and CVs of the poverty incidence for the 5 selected provinces.

```
> results.EB <- data.frame(ProvinceIndex = pbmse.EB$est$eb$domain,
+                          ProvinceName = provincelabels,
+                          SampleSize = pbmse.EB$est$eb$sampsize,
+                          EB = pbmse.EB$est$eb$eb, cv.EB = pbcv.EB)
```

```
> results.EB
  ProvinceIndex ProvinceName SampleSize        EB    cv.EB
1            42        Soria         20 0.2104329 21.06776
2             5        Avila         58 0.1749877 19.49466
3            34     Palencia         72 0.2329916 11.57829
4            44       Teruel         72 0.2786618 11.89621
5            40      Segovia         58 0.2627178 13.21378
```

Other SAE software

The Official Statistics & Survey Methodology CRAN Task View of R (Templ, 2014) contains a subsection called Small Area Estimation with some packages for SAE. The **rsae** package (Schoch, 2011) contains functions for robust fitting of basic unit and area level models and prediction of area means. The current version does not allow for categorical independent variables. The **JoSae** package (Breidenbach, 2011) includes functions for the unit-level EBLUP (Battese et al., 1988) and generalized regression (GREG) estimators (Särndal, 1984) together with estimated variances. Only univariate unit-level models with a simple block-diagonal variance structure are supported. The **hbsae** package (Boonstra, 2012) can be used to obtain estimates based on unit and area level models, where the model fitting can be done by REML or a hierarchical Bayes procedure. In the last case, numerical integration is used to average over the posterior density for the between-area variance.

Recently, several R packages for SAE that are not included in the subsection Small Area Estimation have been developed. The **mme** package (Lopez-Vizcaino et al., 2014) includes functions for SAE under three different multinomial linear mixed models: a model with independent random effects for each category of the response variable, the same model with additional independent time effects, and the same model with correlated time effects. The **saery** package (Esteban et al., 2014) contains functions for obtaining EBLUPs and their MSEs based on the area-level model with time effects introduced by Rao and Yu (1994) using the REML fitting method. The **sae2** package (Fay and Diallo, 2015) also offers functions for SAE under time-series area level models supporting univariate and multivariate applications. They provide EBLUPs based on the Rao-Yu model as well as a modified ("dynamic") version.

Other R software that is not available at CRAN includes the **SAE** package developed within BIAS project (BIAS, 2005), which provides the classical EBLUP and the spatial EBLUP of Petrucci and Salvati (2006). This project includes other methods based on Bayesian spatial models implemented in WinBUGS (Lunn et al., 2000). Preliminary (and not so user friendly) versions of some of the functions contained in the **sae** package described in this paper can be found in the website of the European project SAMPLE (SAMPLE, 2007).

In addition to R software, SAS provides procedures such as `MIXED`, `IML` and hierarchical Bayes `MCMC`, which fit unit and area level models and provide small area estimates together with estimated mean squared errors. See examples of use in Mukhopadhyay and McDowell (2011). Other SAS macros have been developed in the European project EURAREA (EURAREA, 2001). The software includes macros to calculate GREG estimators under a standard linear regression model, the regression synthetic estimator under two different models and the EBLUP using unit and area level models.

Summary

This paper presents the first R package that gathers most basic small area estimation techniques together with more recent and sophisticated methods, such as those for estimation under a FH model with spatial and spatio-temporal correlation or the methods for estimation of nonlinear parameters based on BHF model. The package contains functions for point estimation and also for mean squared error estimation using modern bootstrap techniques. The functions are described and their use is demonstrated through interesting examples, including an example on poverty mapping. Nowadays, we are developing new methods for small area estimation, which will be included in subsequent versions of the **sae** package.

Acknowledgments

We would like to thank the reviewers for their really careful review of the manuscript. Their comments have led to significant improvement of the paper. This work is supported by grants SEJ2007-64500, MTM2012-37077-C02-01 and FP7-SSH-2007-1.

Bibliography

L. Anselin. *Spatial Econometrics. Methods and Models*. Kluwer, Boston, 1988. [p85]

V. Arora and P. Lahiri. On the superiority of the bayesian method over the BLUP in small area estimation problems. *Statistica Sinica*, 7(4):1053–1063, 1997. [p84]

G. E. Battese, R. M. Harter, and W. A. Fuller. An error-components model for prediction of county crop areas using survey and satellite data. *Journal of the American Statistical Association*, 83(401):28–36, 1988. [p82, 89, 90, 91, 96]

BIAS. *Bayesian Methods for Combining Multiple Individual and Aggregate Data Sources in Observational Studies*, 2005. URL http://www.bias-project.org.uk. [p96]

H. J. Boonstra. *hbsae: Hierarchical Bayesian Small Area Estimation*, 2012. URL http://CRAN.R-project.org/package=hbsae. R package version 1.0. [p96]

J. Breidenbach. *JoSAE: Functions for Unit-Level Small Area Estimators and their Variances*, 2011. URL http://CRAN.R-project.org/package=JoSAE. R package version 0.2. [p96]

N. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, 1993. [p85]

G. S. Datta and P. Lahiri. A unified measure of uncertainty of estimated best linear unbiased predictors in small area estimation problems. *Statistica Sinica*, 10(2):613–627, 2000. [p84]

G. S. Datta, J. N. K. Rao, and D. D. Smith. On measuring the variability of small area estimators under a basic area level model. *Biometrika*, 92(1):183–196, 2005. [p84]

M. D. Esteban, D. Morales, and A. Perez. *saery: Small Area Estimation for Rao and Yu Model*, 2014. URL http://CRAN.R-project.org/package=saery. R package version 1.0. [p96]

EURAREA. *Enhancing Small Area Estimation Techniques to Meet European Needs*, 2001. URL http://www.ons.gov.uk/ons/guide-method/method-quality/general-methodology/spatial-analysis-and-modelling/eurarea/index.html. [p96]

R. E. Fay and M. Diallo. *sae2: Small Area Estimation: Time-Series Models*, 2015. URL http://CRAN.R-project.org/package=sae2. R package version 0.1-1. [p96]

R. E. Fay and R. A. Herriot. Estimation of income from small places: An application of James-Stein procedures to census data. *Journal of the American Statistical Association*, 74(366):269–277, 1979. [p82, 83, 84]

W. González-Manteiga, M. J. Lombardía, I. Molina, D. Morales, and L. Santamaría. Bootstrap mean squared error of a small-area EBLUP. *Journal of Statistical Computation and Simulation*, 78(5):443–462, 2008. [p90, 93]

C. R. Henderson. Best linear unbiased estimation and prediction under a selection model. *Biometrics*, 31(2):423–447, 1975. [p83]

R. N. Kackar and D. A. Harville. Approximations for standard errors of estimators of fixes and random effects in mixed linear models. *Journal of the American Statistical Association*, 79(388):853–862, 1984. [p84]

E. Lopez-Vizcaino, M. J. Lombardia, and D. Morales. *mme: Multinomial Mixed Effects Models*, 2014. URL http://CRAN.R-project.org/package=mme. R package version 0.1-5. [p96]

T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(1):1–19, 2004. R package version 2.2. [p84]

T. Lumley. *survey: Analysis of Complex Survey Samples*, 2012. URL http://CRAN.R-project.org/package=survey. R package version 3.28-2. [p84]

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. [p96]

Y. Marhuenda, I. Molina, and D. Morales. Small area estimation with spatio-temporal Fay-Herriot models. *Computational Statistics and Data Analysis*, 58:308–325, 2013. [p87, 88]

I. Molina and J. N. K. Rao. Small area estimation of poverty indicators. *The Canadian Journal of Statistics*, 38(3):369–385, 2010. [p92]

I. Molina, N. Salvati, and M. Pratesi. Bootstrap for estimating the mean squared error of the spatial EBLUP. *Computational Statistics*, 24:441–458, 2009. [p86]

P. Mukhopadhyay and A. McDowell. Small area estimation for survey data analysis using SAS software. In *SAS Global Forum 2011*, 2011. URL http://support.sas.com/resources/papers/proceedings11/336-2011.pdf. [p96]

A. Petrucci and N. Salvati. Small area estimation for spatial correlation in watershed erosion assessment. *Journal of Agricultural, Biological and Environmental Statistics*, 11(2):169–182, 2006. [p85, 96]

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2013. R package version 3.1-111. [p83]

N. G. N. Prasad and J. N. K. Rao. The estimation of the mean squared error of small-area estimators. *Journal of the American Statistical Association*, 85(409):163–171, 1990. [p84]

J. N. K. Rao. *Small Area Estimation*. John Wiley & Sons, 2003. [p81]

J. N. K. Rao and M. Yu. Small area estimation by combining time series and cross-sectional data. *Canadian Journal of Statistics*, 22(4):511–528, 1994. [p96]

R. M. Royall. On finite population sampling theory under certain linear regression. *Biometrika*, 57(2): 377–387, 1970. [p90]

SAMPLE. *Small Area Methods for Poverty and Living Condition Estimates*, 2007. URL http://www.sample-project.eu/. [p96]

C. Särndal. Design-consistent versus model-dependent estimation for small domains. *Journal of the American Statistical Association*, 79(387):624–631, 1984. [p96]

T. Schoch. *rsae: Robust Small Area Estimation*. R package version 0.1-4, 2011. URL http://CRAN.R-project.org/package=rsae. [p96]

B. Singh, G. Shukla, and D. Kundu. Spatio-temporal models in small area estimation. *Survey Methodology*, 31(2):183–195, 2005. [p86]

M. Templ. CRAN task view: Official statistics & survey methodology, 2014. URL http://CRAN.R-project.org/view=OfficialStatistics. Version 2014-08-18. [p96]

Y. Tillé and A. Matei. *sampling: Survey Sampling*, 2012. URL http://CRAN.R-project.org/package=sampling. R package version 2.5. [p84]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. [p83]

Y. You and B. Chapman. Small area estimation using area level models and estimated sampling variances. *Survey Methodology*, 32(1):97–103, 2006. [p84, 85]

Isabel Molina
Department of Statistics
Universidad Carlos III de Madrid
28903 Getafe, Madrid
Instituto de Ciencias Matemáticas (ICMAT)
28049 Madrid
Spain
isabel.molina@uc3m.es

Yolanda Marhuenda
Instituto Centro de Investigación Operativa (CIO)
Department of Statistics, Mathematics and Informatics
Universidad Miguel Hernández de Elche
03202 Elche, Alicante
Spain
y.marhuenda@umh.es

showtext: Using System Fonts in R Graphics

by Yixuan Qiu

Abstract This article introduces the **showtext** package that makes it easy to use system fonts in R graphics. Unlike other methods to embed fonts into graphics, **showtext** converts text into raster images or polygons, and then adds them to the plot canvas. This method produces platform-independent image files that do not rely on the fonts that create them. It supports a large number of font formats and R graphics devices, and meanwhile provides convenient features such as using web fonts and integrating with **knitr**. This article provides an elaborate introduction to the **showtext** package, including its design, usage, and examples.

Introduction

Using fonts in R graphics is neither a new topic nor a difficult task, if only the standard font families such as "sans", "serif" and "mono" are needed. However, problems occur when one wants to select fonts that are installed in the system but not among the standard families inside R, especially for the PDF graphics device. With the evolution of R graphics device as well as related extension packages, there are more and more solutions emerging to solve the font problem. The R News article Murrell and Ripley (2006) systematically describes the working mechanism of PostScript and PDF devices to handle nonstandard fonts, and more recently, the blog post by Winston Chang[1] serves as an tutorial for the **extrafont** package (Chang, 2014) which makes it easy to use TrueType fonts in PostScript, PDF and Windows bitmap devices.

With the same target, this article introduces the **showtext** package (Qiu, 2015a) that provides an alternative way to use fonts in R graphics. It has good support for various font formats and most graphics devices in R, and meanwhile provides some extra features such as loading web fonts and integration with **knitr** (Xie, 2013, 2014a,b). All efforts devoted to the **showtext** package are trying to seek an easy and elegant way to make use of different fonts in R graphics.

In the remaining part, this article will first review some existing methods of font selection in R, and then give an introduction to the **showtext** package, including its design, usage, examples and a number of suggestions for use.

Existing methods to use fonts

Built-in graphics devices in R

It is possible to let R's built-in graphics devices (e.g., PNG and PDF) to use installed fonts in the system. However, the implementation is quite configuration dependent. If Cairo graphics[2] support has been compiled in R such that `png(type = "cairo")` and `cairo_pdf()` are available, then it is quite straightforward to use system fonts in the plots. One only needs to specify the family name of font as is used by the system. (Figure 1)

```
> library(ggplot2)
> # background for various plots later
> bg <- ggplot(NULL, aes(x = 1, y = 1)) + ylim(0.8, 1.2) +
+     theme(axis.title = element_blank(), axis.ticks = element_blank(),
+           axis.text = element_blank())
>
> if(capabilities("cairo")) {
+     png("builtin-1.png", 672, 384, type = "cairo", res = 96)
+     txt1 <- annotate("text", 1, 1, label = "A sample of\nDejaVu Sans Mono",
+                      family = "DejaVu Sans Mono", size = 15)
+     print(bg + txt1)
+     dev.off()
+
+     cairo_pdf("builtin-2.pdf", 7, 4)
```

[1]http://blog.revolutionanalytics.com/2012/09/how-to-use-your-favorite-fonts-in-r-charts.html
[2]http://cairographics.org/

```
+     txt2 <- annotate("text", 1, 1, label = "A sample of\nDejaVu Serif Italic",
+                      family = "DejaVu Serif", fontface = "italic", size = 15)
+     print(bg + txt2)
+     dev.off()
+ }
```

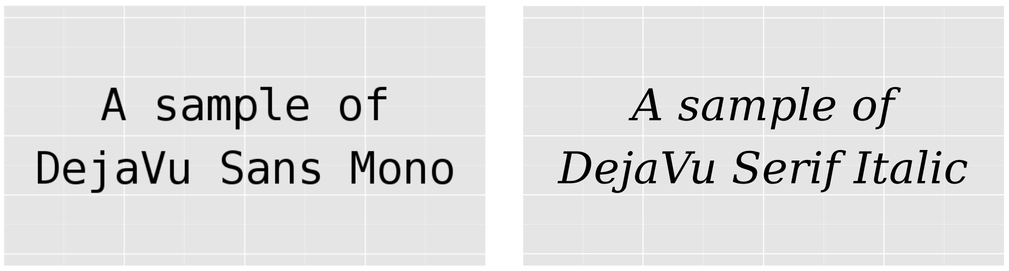


Figure 1: Specify full family name when Cairo graphics is compiled into R. *Left*: graph in PNG format; *right*: graph in PDF format.

However, when Cairo graphics is not available, it will require more effort to customize the font. For PNG graphs, the user needs to first register a font family name in R which is mapped to a font that is installed in the system, and then specify the font family name in plotting functions. Below is an example to show this procedure on Windows. (Figure 2)

```
> if(.Platform$OS.type == "windows") {
+     windowsFonts(century = "Century Gothic")
+     png("builtin-3.png", 672, 384, res = 96)
+     txt3 <- annotate("text", 1, 1, label = "A sample of\nCentury Gothic",
+                      family = "century", size = 20)
+     print(bg + txt3)
+     dev.off()
+ }
```

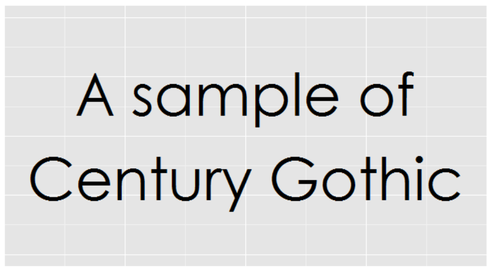


Figure 2: Use system fonts for PNG on Windows without Cairo graphics.

Notice that we use the function `windowsFonts()` to register font and create name mapping. In other operating systems, there are analogous functions such as `X11Fonts()` and `quartzFonts()` to do the similar job.

For PDF graphs, the setup is more complicated. The first step is similar: one should call `pdfFonts()` to register new family names in R, and then use them in the plot. However, the obstacle here is that `pdfFonts()` requires the Adobe Font Metrics files (`.afm`), which may be unavailable for users. On the contrary, TrueType fonts (`.ttf`) and OpenType fonts (mostly `.otf`) are most commonly used, but unfortunately, these font formats are not directly supported by `pdfFonts()`. Additionally, to make the plot have consistent appearance across different PDF viewers, it is usually suggested to embed the font within the file by calling the function `embedFonts()`, which further invokes an external software Ghostscript[3]. For the details, interested readers are referred to the R News article Murrell and Ripley (2006).

Due to this complexity, when creating PDF graphs users are most likely to only select the built-in PDF font families, which can be queried by the command `names(pdfFonts())`.

[3]`http://www.ghostscript.com/`

The Cairo package

The **Cairo** package (Urbanek and Horner, 2014) provides a number of high-quality graphics devices that are driven by a unified back-end based on Cairo graphics. When the Cairo library is configured with FreeType and Fontconfig support, one can make use of the `CairoFonts()` function to specify the fonts that will be used by all the devices in **Cairo**. Below is an example to show this. (Figure 3, left)

```
> library(Cairo)
>
> CairoFonts(regular = "Liberation Sans:style=Regular",
+            italic = "Liberation Serif:style=Italic")
>
> CairoPDF("Cairo-1.pdf", 7, 4)
> txt4 <- annotate("text", 1, 1.1, label = "A sample of Liberation Sans", size = 12)
> txt5 <- annotate("text", 1, 0.9, label = "A sample of Liberation Serif",
+                  fontface = "italic", size = 12)
> print(bg + txt4 + txt5)
> dev.off()
```

For systems where Cairo is not configured with FreeType or Fontconfig (e.g., Windows), the method to specify fonts is similar to that of `png(type = "cairo")` and `cairo_pdf()`. (Figure 3, right)

```
> if(.Platform$OS.type == "windows") {
+     CairoPDF("Cairo-2.pdf", 7, 4)
+     txt6 <- annotate("text", 1, 1.1, label = "A sample of Constantia",
+                      family = "Constantia", size = 12)
+     txt7 <- annotate("text", 1, 0.9, label = "A sample of Lucida Console",
+                      family = "Lucida Console", size = 10)
+     print(bg + txt6 + txt7)
+     dev.off()
+ }
```

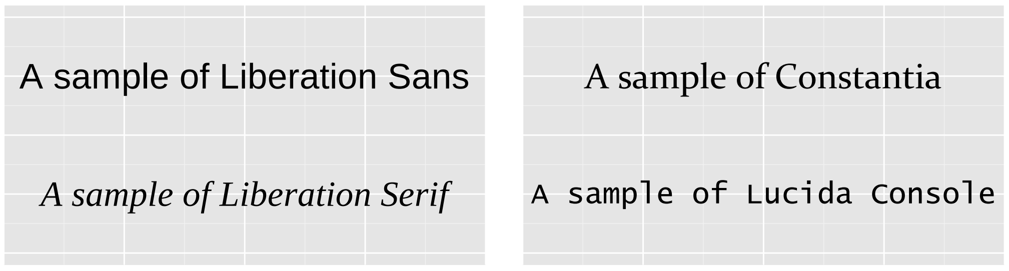


Figure 3: Specify fonts in the **Cairo** package. *Left*: with FreeType and Fontconfig support; *right*: without (e.g. on Windows).

The extrafont package

extrafont is an R package mainly used to simplify the use of system fonts in PDF and PostScript graphics. It is able to extract metric files (`.afm`) from TrueType fonts (`.ttf`) so that the R PDF device can utilize that information to place text in graphics. This procedure is accomplished by the **Rttf2pt1** package (Chang, 2015) that **extrafont** depends on. Also, for the same reason previously described, **extrafont** provides the function `embed_fonts()` to call Ghostscript to embed fonts in PDF files.

extrafont requires a first-time configuration, during which it will scan for TrueType fonts installed in the system and generate corresponding metric files along with other necessary configuration files. Afterwards, one needs to call `loadfonts()` to register the newly created metric fonts in R. These two steps only need to be done once and are not necessary in a new R session.

```
> library(extrafont)
> ## Run once
> font_import()
> loadfonts()
```

After this set-up, the user may query the available font families for the PDF device by calling functions `fonts()` or `fonttable()`, and use them in any plotting functions involving font selection. Since we are now using metric font files, it is best practice to embed the fonts into the PDF file. This is done by calling the function `embed_fonts()`. (Figure 4)

```
> library(extrafont)
> pdf("extrafont-1-unembedded.pdf", 7, 4)
> txt8 <- annotate("text", 1, 1.1, label = "A sample of Ubuntu Light",
+                  family = "Ubuntu Light", size = 12)
> txt9 <- annotate("text", 1, 0.9, label = "A sample of Ubuntu Condensed",
+                  family = "Ubuntu Condensed", size = 12)
> print(bg + txt8 + txt9)
> dev.off()
> embed_fonts("extrafont-1-unembedded.pdf", outfile = "extrafont-1.pdf")
```

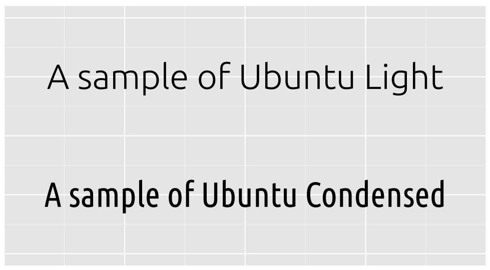


Figure 4: Using the **extrafont** package to select fonts in PDF device.

The embedding procedure requires the Ghostscript program to be installed in the system and findable to R. On Windows, the command below is an example to tell R where it is.

```
> Sys.setenv(R_GSCMD = "C:/Program Files/gs/gs9.05/bin/gswin32c.exe")
```

More details about **extrafont** can be found in the README file[4] of this package.

The new approach: showtext

The previous section describes a number of ways to use system fonts in R graphics. While they could be helpful in many situations when dealing with fonts in R, there is still room for seeking more elegant ways to achieve that target, among which the **showtext** package is one trying to meet such a goal. The following are a number of highlighted features of **showtext**:

1. **Easy installation**. **showtext** only requires the lightweight FreeType library[5] for installation, and works without dependence on external software such as Ghostscript. This would be helpful when other solutions are not possible, e.g., when the Cairo library is unavailable in the system.
2. **Support for various font formats**. A certain type of font is supported as long as the back-end FreeType library can read it, including but not limited to TrueType fonts, OpenType fonts, Type 1 fonts, etc.
3. **Support for various R graphics devices**. Technically **showtext** can work on almost any graphics device, regardless of if it is in PNG, PDF, SVG or JPEG format.
4. The output graph produced by **showtext** has a **platform-independent appearance** . There is no need to embed fonts into the graph, and viewers can read the text without installing the fonts that actually produced them.
5. It also features functions to **automatically search and download** many beautiful, accessible and open source fonts on the web, and users can use these fonts without installing them to the operating system, which means that the system level font library can be kept intact and clean.
6. **showtext** has a neat **integration with knitr**.

[4]http://CRAN.R-project.org/package=extrafont/README.html
[5]http://www.freetype.org/

The basic idea behind **showtext** is quite simple: it converts text into raster images (for bitmap and on-screen devices) or polygons (for vector graphics), and then put them in the graphics. This design comes from the fact that handling text is a very complicated task for graphics devices, but polygons and raster images are lower-level operations that are easier to deal with. Therefore, as long as a device understands how to draw polygons or overlay bitmap on its canvas, it will also be able to show text with the help of this package.

More specifically, the **showtext** package develops a general framework to render text in R graphics. First, it overrides the functions contained in the graphics device that are responsible for drawing text, so that **showtext** will take over the text rendering procedure. Then, it uses the FreeType library to read the font file and analyze the character string that is going to be displayed in the graph. Finally, the text is transformed into basic graphical components (raster images or polygons) that can be easily rendered by the device. As a result, the created graph does not rely on the original font file, thus being platform-independent.

This procedure can be better explained by the diagram in Figure 5.

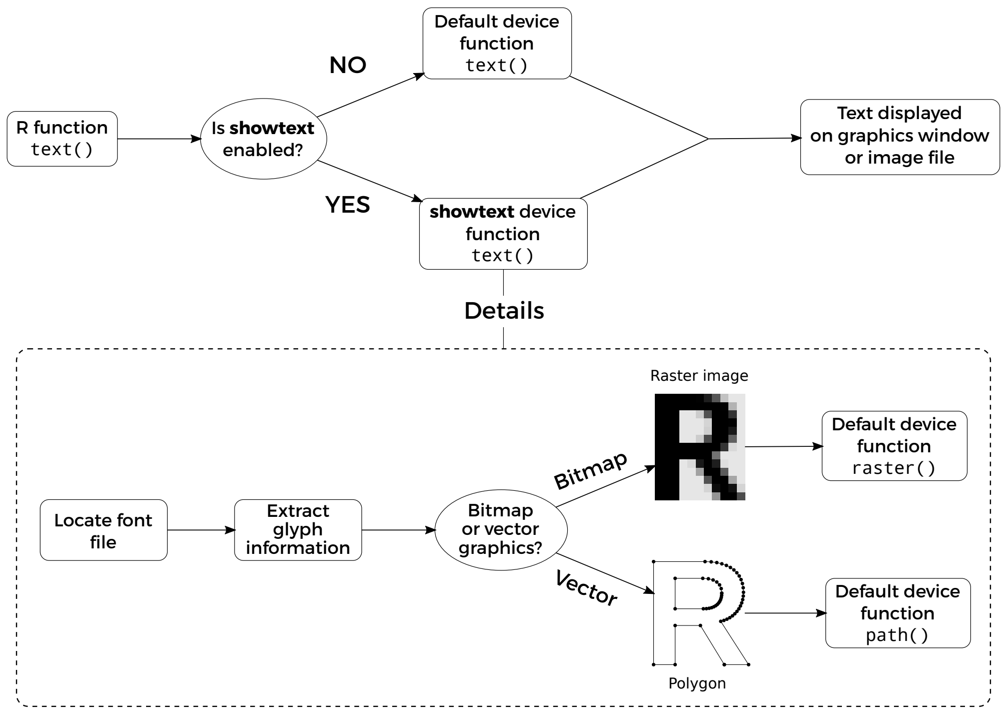


Figure 5: How **showtext** works with R graphics devices.

Usage of showtext

The usage of **showtext** is easy and intuitive, consisting of two major steps: registering system fonts into R, and enabling **showtext** when executing plotting commands. As an additional feature, its integration with **knitr** is also introduced in this section.

Registering fonts

The purpose of this step is to create a mapping between the font family name used by R and the path of the corresponding font file, so that every time the graphics device requests a font with a given name, **showtext** can locate and open that font file. The actual work of registering fonts is done by the **sysfonts** package (Qiu, 2015b), on which **showtext** depends. **sysfonts** provides the function `font.add()` to register font families for **showtext**, with six arguments in total, of which the first two are mandatory. The first parameter, `family`, is the family name that the user wants to use in the plotting functions. The second one, `regular`, should give the path to the font file for a regular font face. Other parameters, such as `bold` and `italic`, are similar to `regular`, but pointing to the files with corresponding font faces. If any of the extra font face parameter is set to `NULL`, the font file for regular font face will be used.

Below is an example to download and register the xkcd[6] font for **showtext**.

```
> library(showtext)
> dest <- file.path(tempdir(), "xkcd.ttf")
> download.file("http://simonsoftware.se/other/xkcd.ttf", dest, mode = "wb")
> font.add("myxkcd", regular = dest)
```

If successful, "myxkcd" should appear in the result returned by `font.families()`. This function lists all font families that are available in **showtext**, among which "sans", "serif", "mono" and "wqy-microhei" are built-in and will be loaded automatically with the package.

For most operating systems, fonts are usually installed in some standard locations. To add fonts located in these directories, users can provide the filename rather than the absolute path to save some typing. For example on Windows, **sysfonts** knows about the standard font directory, so we can use the following code to register the Consolas families to **showtext**:

```
> if(.Platform$OS.type == "windows") {
+     font.add("consolas", regular = "consola.ttf", bold = "consolab.ttf",
+          italic = "consolai.ttf", bolditalic = "consolaz.ttf")
+     font.families()
+ }

[1] "sans"          "serif"        "mono"         "wqy-microhei"
[5] "myxkcd"        "consolas"
```

Users can view or set such search paths through function `font.paths()`, and list available font files in those paths by calling `font.files()`. While it may take some efforts to figure out the filename of a font with a given family name (and perhaps also font face), the naming convention of font files is usually intuitive and fixed. Also, some font file viewers can help mapping the font name to its real file name in the system.

To make the font adding process easier, **sysfonts** also makes use of the Google Fonts project[7] to simplify the process of downloading and registering fonts available on the web. Google Fonts hosts more than 600 open source fonts, and is still enriching its collection. Function `font.families.google()` lists the presently accessible fonts in the repository, and `font.add.google()` could search for a specific font family, download its font files for all possible faces, and add them to **showtext**. These two functions require the **RCurl** (Temple Lang, 2015) and **jsonlite** (Ooms, 2014) packages. The following code demonstrates this process.

```
> head(font.families.google(), 10)

 [1] "ABeeZee"         "Abel"             "Abril Fatface"
 [4] "Aclonica"        "Acme"             "Actor"
 [7] "Adamina"         "Advent Pro"       "Aguafina Script"
[10] "Akronim"

> font.add.google("Lato", "lato")
> font.families()

[1] "sans"          "serif"        "mono"         "wqy-microhei"
[5] "myxkcd"        "consolas"     "lato"
```

Note that the Lato font has multiple font faces (regular, bold, italic, etc.) in the Google Fonts repository. `font.add.google()` is aware of this and will register all of the font faces in R.

Enabling showtext in plots

After registering the fonts, using them in plotting functions will be straightforward. The simplest way to enable **showtext** in R graphs is to call the `showtext.auto()` function, after which users are allowed to use the font families that are listed in `font.families()` to draw text. When it is no longer needed, users may call `showtext.auto(FALSE)` to turn **showtext** off. Figure 6 is an example to demonstrate this usage with **ggplot2** (Wickham 2009; credit goes to the answer in Stackoverflow[8] for inserting an image in **ggplot2**):

[6] http://xkcd.com/
[7] http://www.google.com/fonts
[8] http://stackoverflow.com/questions/9917049/inserting-an-image-to-ggplot2

```
> library(png)
> library(grid)
> library(ggplot2)
> ## download and read an image
> dest <- file.path(tempdir(), "pic.png")
> download.file("http://china-r.org/img/China-R-Logo-trans.png", dest, mode = "wb")
> g <- rasterGrob(readPNG(dest), interpolate = TRUE)
> ## load font and plot
> font.add.google("Lato", "lato")
> ttl <- "\u6b22\u8fce\u5173\u6ce8\u4e2d\u56fd\u0052\u8bed\u8a00\u4f1a\u8bae"
> plt <- ggplot(NULL, aes(x = 1, y = 1)) + xlim(73, 135) + ylim(17, 54) +
+     annotation_custom(g, xmin = 73, xmax = 135, ymin = 17, ymax = 54) +
+     annotate("text", -Inf, -Inf, label = "http://china-r.org", size = 8,
+              family = "lato", fontface = "italic", hjust = -0.1, vjust = -1) +
+     coord_fixed() + ggtitle(ttl) + theme_grey(base_size = 20) +
+     theme(axis.title = element_blank(),
+           plot.title = element_text(family = "wqy-microhei"))
>
> showtext.opts(dpi = 96)
> showtext.auto()
>
> ggsave("showtext-1.png", plt, width = 8.75, height = 5, dpi = 96)
```

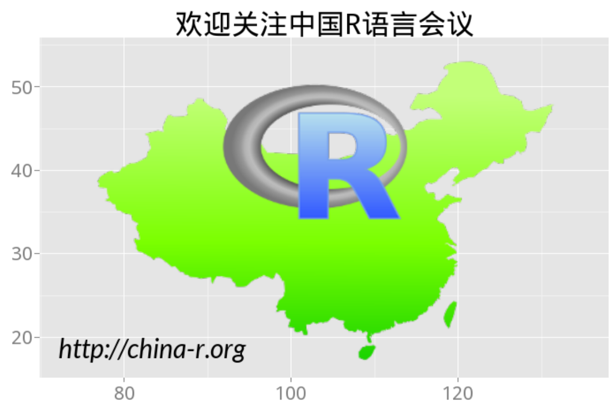


Figure 6: Use **showtext** with **ggplot2**.

In this example, "wqy-microhei" is the name for the WenQuanYi Micro Hei[9] font that will be automatically loaded by **showtext**. WenQuanYi Micro Hei contains a large number of CJK (**C**hinese, **J**apanese and **K**orean) characters, so combined with **showtext** it can be useful to show text in those languages.

Note that when working with bitmap image formats (e.g. PNG, JPEG, TIFF), users should be careful about the resolution of the image. Since **showtext** is unable to query the DPI that is used by the graphics device, users should set it manually by the command `showtext.opts(dpi = ...)`.

While `showtext.auto()` should be enough for most cases in using **showtext**, users actually have more freedom to control which part of the graph should be rendered by **showtext** and which not. Generally speaking, users could enclose code that wants to use **showtext** by a pair of function calls: `showtext.begin()` and `showtext.end()`. The code outside of these parts will still use the standard way to draw text. Figure 7 is an example to show this.

```
> ## load font
> dest <- file.path(tempdir(), "xkcd.ttf")
> download.file("http://simonsoftware.se/other/xkcd.ttf", dest, mode = "wb")
> font.add("myxkcd", regular = dest)
>
> pdf("showtext-2.pdf", 7, 3)
>
> set.seed(0)
```

[9]http://wenq.org/en

```
> p <- runif(1)
> showtext.begin()
> op <- par(family = "myxkcd", mar = c(0.1, 0.1, 3.1, 1.1))
> pie(c(1 - p, p), cex = 1.2, labels = c("Those who understand\nbinary",
+                                         "Those who don't"),
+     col = c("#F8766D", "#00BFC4"), border = NA, radius = 0.9)
> box()
> par(op)
> showtext.end()
> title("There are 10 types of people in the world", font.main = 4)
>
> dev.off()
```

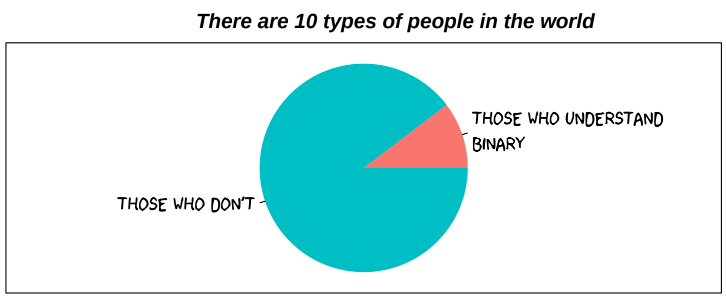


Figure 7: Use **showtext** in part of the graph. The title is drawn using a standard font, while the pie chart labels are using fonts loaded by **showtext**.

Figure 7 uses the xkcd font we added in the previous section. More interesting graphs of this style can be found in the **xkcd** package (Manzanera, 2014).

Integration with knitr

knitr is an R package and engine to generate dynamic documents with R. It is similar to the built-in Sweave engine inside R, but brings in many extensions and enhancements. Starting from version 1.7, **knitr** began to support **showtext** through the option `fig.showtext`. Code chunks with this option being `TRUE` will automatically invoke the `showtext.begin()` function, so there is no need to manually call it from the user. Here is a minimal example of an Rmd file that uses **showtext**.

~~~
We first do some setup work...

```{r setup}
library(knitr)
library(showtext)
showtext.opts(dpi = 72)
opts_chunk$set(fig.width = 7, fig.height = 7, dpi = 72)
```

Then register a font from Google Fonts.

```{r fonts, message=FALSE}
font.add.google("Lobster", "lobster")
```

Finally we create some fancy plot.

```{r fancy, fig.showtext=TRUE, fig.align='center'}
plot(1, pch = 16, cex = 3)
text(1, 1.1, "A fancy dot", family = "lobster", col = "steelblue", cex = 3)
```
~~~

Limitations and solutions

The goal of **showtext** is to provide an elegant way to allow R to use system fonts in graphics. While it should be useful in most situations, there are a few limitations that need to be taken care of. This section lists these limitations of **showtext**, as well as some hints about how to use it in the best way.

First, **showtext** looks for fonts according to their filenames rather than the usual "font names" in the system. This design is intentional since font names can be ambiguous. For example, the same font can have multiple names given by different font management software. In contrast, the font file is the entity that actually contains the glyphs of a specific font, hence it helps to avoid such confusion. While this setting may cause some problems for users who are searching for the filename for the first time, it should be quite convenient afterwards, and one possible solution is to use the fonts in Google Fonts through the function `font.add.google()`, which maintains a standard and stable interface to access fonts.

Second, for vector graphics such as PDF and SVG, since text will be converted into polygons, it is no longer real text that can be searched in a PDF or SVG viewer. Also, the size of the PDF or SVG file created by **showtext** is usually larger than the one produced in the standard way. For users concerned by these issues, it is advisable touse `cairo_pdf()` or **Cairo** to generate PDF graphics, and to use **RSvgDevice** (Luciani et al., 2014) for SVG output.

In addition, at the time of writing **showtext** is not working well with the plot window provided by the RStudio IDE[10]. The simple solution is to manually open a graphics device using functions such as `windows()` and `x11()`, and then draw plots inside this window, rather than the built-in one offered by RStudio IDE.

Finally, in terms of the overall design, **showtext** is in some sense intrusive, since when user calls `showtext.begin()`, it temporarily replaces the device functions by its own ones, and later restores them after `showtext.end()` is called. An alternative and probably more elegant way to address the font problem is to develop new graphics devices that are based on **showtext**, so that they can make use of the text handling functionality contained in **showtext**. This possibility is left to future developers of graphics devices.

Summary

This article introduces the **showtext** package, which helps to use various types of fonts in R graphics, with both highlighted features and its limitations. We have also discussed a number of alternative ways in R to create graphs using non-standard fonts. While there is hardly a universally best way to use fonts in R, some situations were described in which a certain approach is most appropriate.

- The built-in devices in R without Cairo support have the least dependency on external libraries and software. More effort needs to be taken in this situation compared with others, but this may be the only possible solution, especially when R is compiled in a minimal environment.
- When Cairo graphics is compiled in R, devices such as `png(type = "cairo")` and `cairo_pdf()` allow users to select fonts by their family names. This should be the easiest way to use fonts in R graphs without extension packages.
- The **Cairo** package has similar functionality to `png(type = "cairo")` and `cairo_pdf()`, and additionally provides a global font selector, which is useful when choosing a default font for all **Cairo** devices.
- The **extrafont** package provides a complete solution for `pdf()` and `postscript()` devices with TrueType fonts, as well as support for Windows bitmap output. It introduces an easy way to import system fonts into R, so that users only need to make configuration changes once.
- **showtext** supports various font formats and most graphics devices. It also offers some convenient features like using web fonts obtained from Google Fonts, and integration with the **knitr** package. Some suggestions are given in Section "Limitations and solutions" in order to use it in the best way.

Acknowledgments

The author of this article would like to thank Yihui Xie, for his kind work of adding the support for **showtext** in his **knitr** package, and Barret Schloerke, who provided a lot of helpful comments and corrections of the original manuscript. The author is also grateful to the editor and two anonymous reviewers, who made many helpful suggestions to this article.

[10]http://www.rstudio.com/products/RStudio/

Bibliography

W. Chang. *extrafont: Tools for Using Fonts*, 2014. URL http://CRAN.R-project.org/package=extrafont. R package version 0.17. [p99]

W. Chang. *Rttf2pt1: Package for ttf2pt1 Program*, 2015. URL http://CRAN.R-project.org/package=Rttf2pt1. R package version 1.3.3. [p101]

T. J. Luciani, M. Decorde, and V. Lise. *RSvgDevice: An R SVG Graphics Device*, 2014. URL http://CRAN.R-project.org/package=RSvgDevice. R package version 0.6.4.4. [p107]

E. T. Manzanera. *xkcd: Plotting ggplot2 Graphics in a XKCD Style*, 2014. URL http://CRAN.R-project.org/package=xkcd. R package version 0.0.4. [p106]

P. Murrell and B. Ripley. Non-standard fonts in PostScript and PDF graphics. *R News*, 6(2):41–47, May 2006. URL http://CRAN.R-project.org/doc/Rnews/. [p99, 100]

J. Ooms. The jsonlite package: A practical and consistent mapping between JSON data and R objects. *arXiv:1403.2805 [stat.CO]*, 2014. URL http://arxiv.org/abs/1403.2805. [p104]

Y. Qiu. *showtext: Using Fonts More Easily in R Graphs*, 2015a. URL http://CRAN.R-project.org/package=showtext. R package version 0.4-2. [p99]

Y. Qiu. *sysfonts: Loading System Fonts into R*, 2015b. URL http://CRAN.R-project.org/package=sysfonts. R package version 0.5. [p103]

D. Temple Lang. *RCurl: General Network (HTTP/FTP/...) Client Interface for R*, 2015. URL http://CRAN.R-project.org/package=RCurl. R package version 1.95-4.6. [p104]

S. Urbanek and J. Horner. *Cairo: R Graphics Device using Cairo Graphics Library for Creating High-Quality Bitmap (PNG, JPEG, TIFF), Vector (PDF, SVG, PostScript) and Display (X11 and Win32) Output*, 2014. URL http://CRAN.R-project.org/package=Cairo. R package version 1.5-6. [p101]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2009. URL http://had.co.nz/ggplot2/book. [p104, 125]

Y. Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, 2013. URL http://CRAN.R-project.org/package=knitr. [p99]

Y. Xie. knitr: A comprehensive tool for reproducible research in R. In V. Stodden, F. Leisch, and R. D. Peng, editors, *Implementing Reproducible Computational Research*. Chapman and Hall/CRC, 2014a. [p99]

Y. Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2014b. URL http://CRAN.R-project.org/package=knitr. R package version 1.6.10. [p28, 99]

Yixuan Qiu
Purdue University
Department of Statistics, West Lafayette, IN 47906
USA
yixuanq@gmail.com

Correspondence Analysis on Generalised Aggregated Lexical Tables (CA-GALT) in the FactoMineR Package

by Belchin Kostov, Mónica Bécue-Bertaut and François Husson

Abstract Correspondence analysis on generalised aggregated lexical tables (CA-GALT) is a method that generalizes classical CA-ALT to the case of several quantitative, categorical and mixed variables. It aims to establish a typology of the external variables and a typology of the events from their mutual relationships. In order to do so, the influence of external variables on the lexical choices is untangled cancelling the associations among them, and to avoid the instability issued from multicollinearity, they are substituted by their principal components. The `CaGalt` function, implemented in the **FactoMineR** package, provides numerous numerical and graphical outputs. Confidence ellipses are also provided to validate and improve the representation of words and variables. Although this methodology was developed mainly to give an answer to the problem of analyzing open-ended questions, it can be applied to any kind of frequency/contingency table with external variables.

Introduction

Frequency tables are a common data structure in very different domains such as ecology (specie abundance table), textual analysis (documents $\times$ words table) and public information systems (administrative register such as mortality data). This type of table counts the occurrences of a series of events (species, words, death causes) observed on different units (ecological sites, documents, administrative areas). Correspondence analysis (CA) is a reference method to analyse this type of tables offering the visualization of the similarities between events, the similarities between units and the associations between events and units (Benzécri, 1973; Lebart et al., 1998; Murtagh, 2005; Greenacre, 2007; Beh and Lombardo, 2014). However, this method presents two main drawbacks, when the frequency table is very sparse:

1. The first axes frequently show the relationships between small sets of units and small sets of events and do not reveal global trends.
2. The interpretation of the similarities/oppositions among units cannot be understood without taking into account the unit characteristics (such as, for example, climatic conditions, socio-economic description of the respondents or economic characteristics of the area).

In order to solve these drawbacks, contextual variables are also observed on the units and introduced in the analysis. A first step consists of grouping the units depending on one categorical variable and building an aggregated frequency table (AFT) crossing the categories (rows) and the events (columns). In this AFT, the former row-units corresponding to the same category are now collapsed into a single row while the event-columns remain unchanged. Then, CA is applied on this AFT, often called, in textual analysis, aggregated lexical table (ALT; Lebart et al., 1998).

CA on the aggregated lexical table (CA-ALT) usually leads to robust and interpretable results. CA-ALT visualizes the similarities among categories, the similarities among words and the associations between categories and words. The same approach can be applied in other domains. The main drawback of CA-ALT is its restrictiveness. Only one categorical variable can be considered while often several categorical and quantitative contextual variables are available and associated to the events.

Recently, correspondence analysis on generalised aggregated lexical tables (CA-GALT; Bécue-Bertaut and Pagès, 2015; Bécue-Bertaut et al., 2014) has been proposed to generalize CA-ALT to the case of several quantitative, categorical and mixed variables. CA-GALT brings out the relationships between the vocabulary and the several selected contextual variables.

This article presents an R function implementing CA-GALT in the **FactoMineR** package (Lê et al., 2008; Husson et al., 2010) and has the following outline: We first describe the example used to illustrate the method and introduce the notation. Then we recall the principles of the CA-GALT methodology and proceed to detail the function and the algorithm. Subsequently, the results obtained on the example are provided. Finally, we conclude with some remarks.

Example

The example is extracted from a survey intended to better know the definitions of health that the non-experts give. An open-ended question "*What does health mean to you?*" was asked to 392 respondents who answered through free-text comments. The documents $\times$ words table is built keeping only the words used at least 10 times among all respondents. This minimum threshold is used to obtain statistically interpretable results (Lebart et al., 1998; Murtagh, 2005). Thus, 115 different words and 7751 occurrences are kept.

The respondents' characteristics are also collected. In this example, we use age in groups (under 21, 21–35, 36–50 and over 50), gender (man and woman) and health condition (poor, fair, good and very good health) as they possibly condition the respondents' viewpoint.

CA-GALT is able to determine the main dispersion dimensions as much as they are related to the respondents' characteristics.

Notation

The data is coded into two matrices (see Figure 1). The $(I \times J)$ matrix $\mathbf{Y}$, with generic term y_{ij}, contains the frequency of the J words in the I respondents' answers. The $(I \times K)$ matrix $\mathbf{X}$, with generic term x_{ik}, stores the K respondents' characteristics, codified as dummy variables from the L categorical variables.

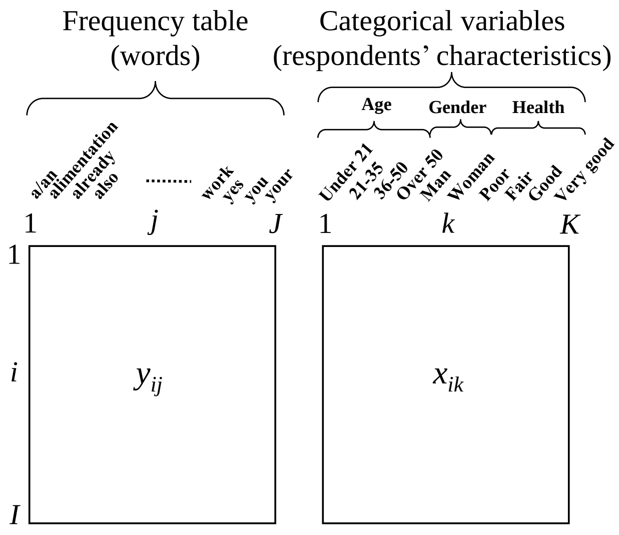


Figure 1: The data set. On the left, the frequency table $\mathbf{Y}$; on the right the categorical table $\mathbf{X}$. In the example, $I = 392$ (respondents), $J = 115$ (words), $K = 10$ (categories).

The proportion matrix $\mathbf{P}$ is computed as $\mathbf{P} = \mathbf{Y}/N$ with generic term $p_{ij} = y_{ij}/N$. The row margin (respectively, column margin) with generic term $p_{i\bullet} = \sum_j p_{ij}$ (respectively, $p_{\bullet j} = \sum_i p_{ij}$) is stored in the $(I \times I)$ diagonal matrix $\mathbf{D}_I$ (respectively, the $(J \times J)$ diagonal matrix $\mathbf{D}_J$). From $\mathbf{P}$, the $(I \times J)$ matrix $\mathbf{Q}$ is defined as $\mathbf{Q} = \mathbf{D}_I^{-1}\mathbf{P}\mathbf{D}_J^{-1}$ with generic term $q_{ij} = p_{ij}/(p_{i\bullet}p_{\bullet j})$. This matrix, or equivalently, its doubly centred form, the $(I \times J)$ matrix $\bar{\mathbf{Q}}$ with generic term $\bar{q}_{ij} = (p_{ij} - p_{i\bullet}p_{\bullet j})/(p_{i\bullet}p_{\bullet j})$ is the matrix analysed by CA. $\bar{\mathbf{Q}}$ evidences that CA analyses the weighted deviation between $\mathbf{P}$ and the $(I \times J)$ independence model matrix $[p_{i\bullet}p_{\bullet j}]$.

The principal component analysis (PCA) applied to a data matrix $\mathbf{Z}$ with metric $\mathbf{M}/\mathbf{D}$ and weighting system $\mathbf{D}/\mathbf{M}$ in the row/column space is noted PCA$(\mathbf{Z}, \mathbf{M}, \mathbf{D})$.

Methodology

The CA-GALT method is detailed in Bécue-Bertaut and Pagès (2015) for quantitative contextual variables and Bécue-Bertaut et al. (2014) for categorical variables. We recall here the principles of this method. To ease the presentation, we recall first that classical CA is a double projected analysis. Then, we show that CA-GALT maintains a similar approach when several quantitative or categorical variables are taken into account.

Classical CA as a particular PCA

It is established that classical CA($\mathbf{Y}$) is equivalent to PCA($\mathbf{Q}, \mathbf{D}_J, \mathbf{D}_I$) or to PCA($\bar{\mathbf{Q}}, \mathbf{D}_J, \mathbf{D}_I$) (Escofier and Pagès, 1988; Böckenholt and Takane, 1994; Bécue-Bertaut and Pagès, 2004). This point of view presents the advantage of placing CA rationale in the general scheme for the principal components methods.

CA of an aggregated lexical table

In this section, the columns of $\mathbf{X}$ are dummy variables corresponding to the categories of a single categorical variable. First, the $(J \times K)$ aggregated lexical table

$$\mathbf{Y}_A = \mathbf{Y}^T\mathbf{X} \tag{1}$$

is built crossing the words and the categories of the categorical variable. Then the $(J \times K)$ proportion matrix is computed as

$$\mathbf{P}_A = \mathbf{P}^T\mathbf{X}. \tag{2}$$

The $(J \times J)$ diagonal matrix

$$\mathbf{D}_J = [d_{Jjj}] = [p_{Aj\bullet}] = [p_{\bullet j}] \tag{3}$$

and the $(K \times K)$ diagonal matrix

$$\mathbf{D}_K = [d_{Kkk}] = [p_{A\bullet k}] \tag{4}$$

store, respectively, the row and colum margins of $\mathbf{P}_A$. $\mathbf{D}_J$ (respectively, $\mathbf{D}_K$) corresponds to weighting the system on the rows (respectively, on the columns). As a single categorical variable is considered

$$\mathbf{D}_K = \mathbf{X}^T\mathbf{D}_I\mathbf{X}. \tag{5}$$

CA-ALT, that is CA($\mathbf{Y_A}$), is performed through PCA($\mathbf{Q}_A, \mathbf{D}_K, \mathbf{D}_J$) where the $(J \times K)$ matrix

$$\mathbf{Q}_A = \mathbf{D}_J^{-1}\mathbf{P}_A\mathbf{D}_K^{-1} \tag{6}$$

is the double standardized form of $\mathbf{P}_A$.

PCA($\mathbf{Q}_A, \mathbf{D}_K, \mathbf{D}_J$) analyses both the dispersion of the cloud of category profiles insofar as explained by the dispersion of the cloud of word profiles and the dispersion of the word profiles insofar as explained by the dispersion of categories profiles. In other words, CA-ALT, as a double projected analysis, allows for the variability of the rows to be explained in terms of the columns and the variability of the columns in terms of the rows (Bécue-Bertaut et al., 2014).

Correspondence analysis on generalised aggregated lexical tables (CA-GALT)

In this section, $\mathbf{X}$ includes the centred dummy columns corresponding to several categorical variables. The aggregated lexical tables built from each categorical variable are juxtaposed row-wise into the generalised aggregated lexical table (GALT) $\mathbf{Y}_A$ of dimensions $J \times K$ (see Figure 2).

In the following, we use the same notation that this used in the former section to highlight that the same rationale is followed regardless the differences existing between the $\mathbf{X}$ structure in both cases. As in the former section

$$\mathbf{Y}_A = \mathbf{Y}^T\mathbf{X}. \tag{7}$$

$\mathbf{Y}_A$ is transformed into the matrix

$$\mathbf{P}_A = \mathbf{Y}_A/N. \tag{8}$$

To maintain a double projected analysis, $\mathbf{D}_K$ is substituted by the $(K \times K)$ covariance matrix $\mathbf{C} = (\mathbf{X}^T\mathbf{D}_I\mathbf{X})$. As $\mathbf{C}$ is not invertible, the Moore-Penrose pseudoinverse $\mathbf{C}^-$ is used and the former Eq. (6) becomes

$$\mathbf{Q}_A = \mathbf{D}_J^{-1}\mathbf{P}_A\mathbf{C}^- \tag{9}$$

of dimensions $J \times K$. CA-GALT is performed through PCA($\mathbf{Q}_A, \mathbf{C}, \mathbf{D}_J$). As in any PCA, the eigenvalues are stored into the $(S \times S)$ diagonal matrix $\mathbf{\Lambda}$, and the eigenvectors into the $(K \times S)$ matrix $\mathbf{U}$. The coordinates of the row-words are computed as

$$\mathbf{F} = \mathbf{Q}_A\mathbf{C}\mathbf{U} \tag{10}$$

and the column-categories as

$$\mathbf{G} = \mathbf{Q}_A^T\mathbf{D}_J\mathbf{F}\mathbf{\Lambda}^{-1/2}. \tag{11}$$

The respondents can be reintroduced in the analysis by positioning the columns of $\mathbf{Q}$ as supplementary

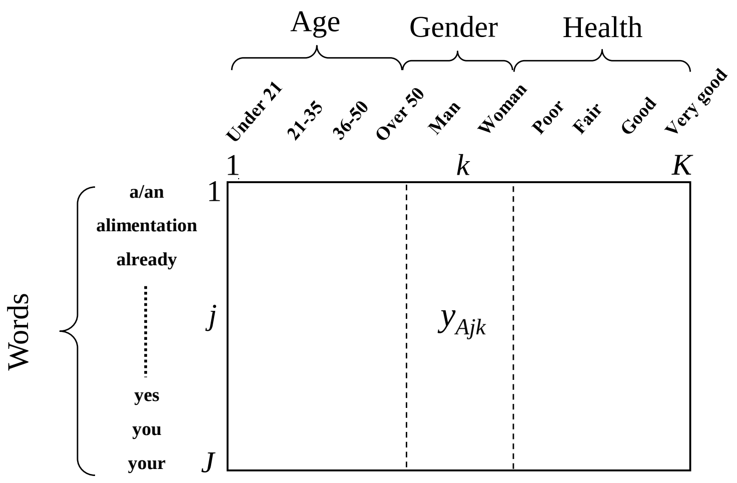


Figure 2: Generalised aggregated lexical table with the words in rows and the categories of age, gender and health condition in columns.

columns in PCA($\mathbf{Q}_A, \mathbf{C}, \mathbf{D}_J$). So the respondents are placed on the axes at the weighted centroid, up to a constant, of the words that they use. Thus, their coordinates are computed via the transition relationships as

$$\mathbf{G}^{+} = \mathbf{D}_I^{-1}\mathbf{P}\mathbf{F}\mathbf{\Lambda}^{-1/2}. \tag{12}$$

The interpretation rules of the results of this specific CA are the usual CA interpretation rules (Greenacre, 1984; Escofier and Pagès, 1988; Lebart et al., 1998).

Quantitative contextual variables

The case of the quantitative variables is detailed in Bécue-Bertaut and Pagès (2015). The same rationale is followed, but defining:

- The GALT $\mathbf{Y}_A$ is built as $\mathbf{Y}_A = \mathbf{Y}^T\mathbf{X}$ and transformed into the matrix $\mathbf{P}_A = \mathbf{P}^T\mathbf{X}$ whose generic term $p_{Ajk} = \sum_i p_{ij}x_{ik}$ is equal to the weighted sum of the values assumed for variable k by the respondents who used word j.
- The matrix $\mathbf{D}_K^{-1}$ no longer exists because the column-margins of $\mathbf{P}_A$ do not correspond to a weighting system on the columns.

Other aspects

- **CA-GALT on principal components**: To obtain less time consuming computation, the original variables are substituted by their principal components, computed either by PCA if $\mathbf{X}$ stores quantitative variables or multiple correspondence analysis if $\mathbf{X}$ stores categorical variables. The components associated to the low eigenvalues can be discarded to solve the instability problem issued from multicolinearity.
- **Confidence ellipses**: To validate and to better interpret the representation of the words and variables, confidence ellipses based on the bootstrap principles (Efron, 1979) are performed.

Using the `CaGalt` function in R

The R function `CaGalt` is currently part of the **FactoMineR** package. The default input for the `CaGalt` function in R is

```
CaGalt(Y, X, type = "s", conf.ellip = FALSE, nb.ellip = 100, level.ventil = 0,
  sx = NULL, graph = TRUE, axes = c(1, 2))
```

with the following arguments:

- `Y`: a data frame with n rows (individuals) and p columns (frequencies)
- `X`: a data frame with n rows (individuals) and k columns (quantitative or categorical variables)

- `type`: the type of variables: `"c"` or `"s"` for quantitative variables and `"n"` for categorical variables. The difference is that for `"s"` variables are scaled to unit variance (by default, variables are scaled to unit variance)
- `conf.ellip`: boolean (`FALSE` by default), if `TRUE`, draw confidence ellipses around the frequencies and the variables when `graph` is `TRUE`
- `nb.ellip`: number of bootstrap samples to compute the confidence ellipses (by default 100)
- `level.ventil`: proportion corresponding to the level under which the category is ventilated; by default, 0 and no ventilation is done. Available only when type is equal to `"n"`
- `sx`: number of principal components kept from the principal axes analysis of the contextual variables (by default is `NULL` and all principal components are kept)
- `graph`: boolean, if `TRUE` a graph is displayed
- `axes`: a length 2 vector specifying the components to plot

The returned value of `CaGalt` is a list containing:

- `eig`: a matrix containing all the eigenvalues, the percentage of variance and the cumulative percentage of variance
- `ind`: a list of matrices containing all the results for the individuals (coordinates, square cosine)
- `freq`: a list of matrices containing all the results for the frequencies (coordinates, square cosine, contributions)
- `quanti.var`: a list of matrices containing all the results for the quantitative variables (coordinates, correlation between variables and axes, square cosine)
- `quali.var`: a list of matrices containing all the results for the categorical variables (coordinates of each categories of each variables, square cosine)
- `ellip`: a list of matrices containing the coordinates of the frequencies and variables for replicated samples from which the confidence ellipses are constructed

To facilitate the analysis of the output of the results and the graphs, corresponding print, plot and summary methods were also implemented in **FactoMineR**.

Application of the `CaGalt` function

To illustrate the outputs and graphs of `CaGalt`, we use the data set presented in the previous section. The first 115 columns correspond to the frequencies of the words in respondents' answers and the last three columns correspond to the categorical variables corresponding to respondents' characteristics (whose type is defined as `"n"`). The code to perform the `CaGalt` is

```
> data(health)
> res.cagalt <- CaGalt(Y = health[, 1:115], X = health[, 116:118], type = "n")
```

Numerical outputs

The results are given in a list for the individuals, the frequencies, the variables and the confidence ellipses.

```
> res.cagalt
**Results for the Correspondence Analysis on Generalised Aggregated Lexical
  Tables (CaGalt)**
*The results are available in the following entries:
   name               description
1  "$eig"             "eigenvalues"
2  "$ind"             "results for the individuals"
3  "$ind$coord"       "coordinates for the individuals"
4  "$ind$cos2"        "cos2 for the individuals"
5  "$freq"            "results for the frequencies"
6  "$freq$coord"      "coordinates for the frequencies"
7  "$freq$cos2"       "cos2 for the frequencies"
8  "$freq$contrib"    "contributions of the frequencies"
9  "$quali.var"       "results for the categorical variables"
10 "$quali.var$coord" "coordinates for the categories"
11 "$quali.var$cos2"  "cos2 for the categories"
```

```
12 "$ellip"          "coordinates to construct confidence ellipses"
13 "$ellip$freq"     "coordinates of the ellipses for the frequencies"
14 "$ellip$var"      "coordinates of the ellipses for the variables"
```

The interpretation of the numerical outputs can be facilitated using the summary method for the objects of class 'CaGalt' returned by function CaGalt which prints summaries of the CaGalt entries.

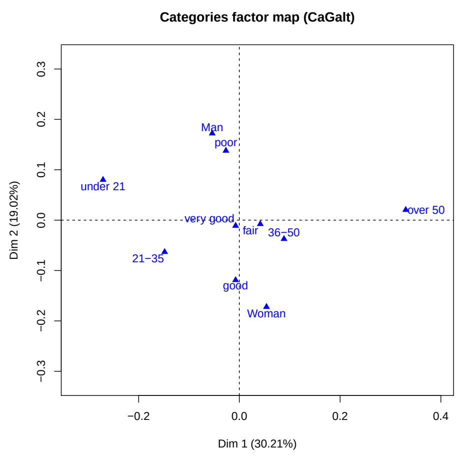
```
> summary(res.cagalt)
Eigenvalues
                      Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
Variance              0.057   0.036   0.026   0.024   0.020   0.013   0.012
% of var.            30.207  19.024  13.776  12.953  10.847   6.819   6.374
Cumulative % of var. 30.207  49.230  63.007  75.960  86.807  93.626 100.000

Individuals (the 10 first individuals)
                Dim.1   cos2    Dim.2   cos2    Dim.3   cos2
6            |  0.120  0.037 | -0.551  0.781 | -0.065  0.011 |
7            | -0.134  0.019 | -0.788  0.649 | -0.166  0.029 |
9            |  0.056  0.002 |  0.272  0.047 | -0.211  0.028 |
10           |  0.015  0.001 | -0.262  0.342 | -0.084  0.035 |
11           | -1.131  0.293 |  0.775  0.138 | -0.613  0.086 |
13           | -0.909  0.231 | -0.340  0.032 |  0.464  0.060 |
14           |  0.097  0.026 |  0.070  0.014 |  0.236  0.154 |
15           | -0.718  0.117 | -1.524  0.526 | -0.717  0.116 |
17           | -0.924  0.372 |  0.074  0.002 |  0.954  0.397 |
18           | -0.202  0.050 |  0.563  0.389 |  0.404  0.200 |

Frequencies (the 10 first most contributed frequencies on the first principal plane)
                   Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
physically      | -0.508  6.062  0.941 | -0.036  0.050  0.005 | -0.045  0.102  0.007 |
to have         |  0.241  5.654  0.727 |  0.054  0.451  0.037 | -0.027  0.157  0.009 |
well            | -0.124  0.790  0.168 | -0.255  5.254  0.703 | -0.073  0.593  0.057 |
to feel         | -0.217  1.174  0.222 | -0.321  4.059  0.484 | -0.158  1.353  0.117 |
hungry          |  0.548  1.504  0.254 | -0.630  3.158  0.336 | -0.367  1.478  0.114 |
I               |  0.360  2.927  0.537 | -0.213  1.623  0.188 | -0.114  0.639  0.053 |
one             |  0.246  0.964  0.241 |  0.369  3.449  0.542 |  0.120  0.500  0.057 |
something       | -0.826  2.959  0.431 |  0.444  1.353  0.124 | -0.734  5.121  0.340 |
best            |  0.669  4.182  0.602 |  0.091  0.123  0.011 |  0.225  1.041  0.068 |
psychologically | -0.369  0.560  0.155 | -0.727  3.444  0.601 |  0.190  0.323  0.041 |

Categorical variables
                Dim.1   cos2    Dim.2   cos2    Dim.3   cos2
21-35        | -0.148  0.347 | -0.063  0.063 |  0.148  0.347 |
36-50        |  0.089  0.108 | -0.037  0.019 |  0.120  0.199 |
over 50      |  0.330  0.788 |  0.020  0.003 | -0.028  0.006 |
under 21     | -0.271  0.484 |  0.080  0.042 | -0.240  0.382 |
Man          | -0.054  0.081 |  0.172  0.826 |  0.018  0.009 |
Woman        |  0.054  0.081 | -0.172  0.826 | -0.018  0.009 |
fair         |  0.042  0.029 | -0.008  0.001 | -0.144  0.342 |
good         | -0.007  0.001 | -0.119  0.185 | -0.077  0.077 |
poor         | -0.027  0.002 |  0.138  0.061 |  0.193  0.120 |
very good    | -0.007  0.000 | -0.011  0.001 |  0.027  0.005 |
```

Regarding the interpretation of these results, the percentage of variance explained by each dimension is given: 30.21% for the first axis and 19.02% for the second one. The third and fourth dimensions also explain an important part of the total variability (13.78% and 12.95%, respectively) and it may be interesting to plot the graph for these two dimensions. The numerical outputs corresponding to the individuals, frequencies and variables are useful especially as a help to interpret the graphical outputs.

The arguments of the summary method for 'CaGalt' objects, nbelements (number of written elements), nb.dec (number of printed decimals) and ncp (number of printed dimensions), can also be modified to obtain more detailed numerical outputs. For more information please check the help file of this function.

Graphical outputs

By default, the `CaGalt` function gives three graphs: one for the individuals, one for the variables and one for the frequencies. The variables graph shows the map of the categories (see Figure 3). The trajectory of *age group* categories notably follows the first axis. The second axis opposes the *Man* category and the *Woman* category. The categories of *health condition* are close to the centroid on the first dimension indicating a low association with the words on this dimension. The categories *poor* and *good* of *health condition* are opposed on the second axis; they lie close to *Man* and *Woman* respectively.



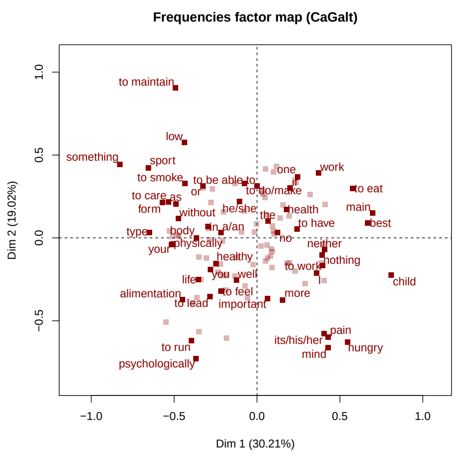


Figure 3: Categories for dimensions 1 and 2. **Figure 4:** Frequencies for dimensions 1 and 2.

Regarding the word graph (see Figure 4), *best*, *main* and *child* are the words with the highest coordinates at the right of the first axis. From the transition relationships between words and categories, we can say that both words are very frequently used by the oldest categories and avoided by the youngest. These words contrast with *something* and *sport*, at the left of the first axis, these latter words being used by the youngest categories and avoided by the oldest. Comparing vocabulary choices depending on the gender (second axis), we see that men more frequently consider health from a physical point of view (*sport*, *to maintain*, and so on) and women from a psychological point of view (*psychologically*, *mind*, and so on).

The high number of elements drawn on the graphs (points and labels) makes the reading and the interpretation difficult. The `plot` method for '`CaGalt`' objects can improve them through replacing the labels, modifying their size, changing the colours, adding confidence ellipses or only selecting a part of the elements. For example, for the categorical variables we will use the code

```
> plot(res.cagalt, choix = "quali.var", conf.ellip = TRUE, axes = c(1, 4))
```

The parameter `choix = "quali.var"` indicates that we plot the graph of the categorical variables, the parameter `axes = c(1,4)` indicates that the graph is done for the dimensions 1 and 4, and the parameter `conf.ellip = TRUE` indicates that confidence ellipses are drawn around the categorical variables. Analyzing the graph which was drawn with this code (see Figure 5), it shows that the fourth axis turns out to be of interest because of ranking health condition categories in their natural order. These categories are well separated, except for the two better health categories whose confidence ellipses overlap.

Another example is provided through the following code

```
> plot(res.cagalt, choix = "freq", cex = 1.5, col.freq = "darkgreen",
+   select = "contrib 10")
```

The ten frequencies (`choix = "freq"`) with highest contributions (`select = "contrib 10"`) are selected and plotted in darkgreen (`col.freq = "darkgreen"`) using a magnification of 1.5 (`cex = 1.5`) relative to the default (see Figure 6).

Conclusion

The main features of the R function `CaGalt` have been detailed and illustrated using an example based on survey data collected by a questionnaire which included an open-ended question. This application to a real data set demonstrates how both open-ended and closed questions combine to provide relevant information. Although this methodology was developed mainly to give an answer to the problem of analyzing open-ended questions with several quantitative, categorical and mixed contextual variables, it can be applied to any kind of frequency/contingency table with external variables. The function `CaGalt` can also be used to perform a CA-ALT because there is no other function in R for the same purpose.

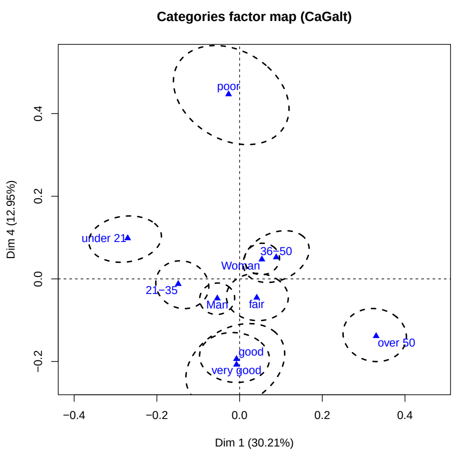


Figure 5: Categories for dimensions 1 and 4 completed by confidence ellipses.

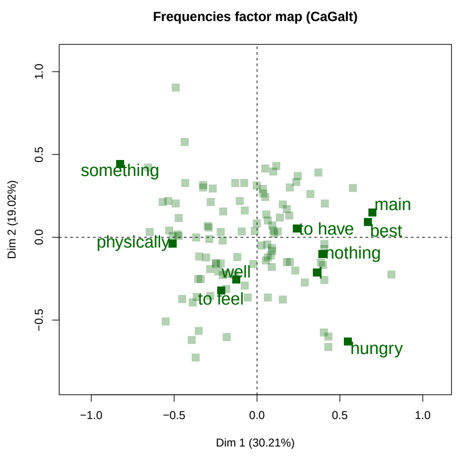


Figure 6: Ten words with the highest contributions on the first principal plane.

Bibliography

M. Bécue-Bertaut and J. Pagès. A principal axes method for comparing multiple contingency tables: MFACT. *Computational Statistics and Data Analysis*, 45(3):481–503, 2004. [p111]

M. Bécue-Bertaut and J. Pagès. Correspondence analysis of textual data involving contextual information: CA-GALT on principal components. *Advances in Data Analysis and Classification*, 9(2):125–142, 2015. [p109, 110, 112]

M. Bécue-Bertaut, J. Pages, and B. Kostov. Untangling the influence of several contextual variables on the respondents' lexical choices. A statistical approach. *SORT – Statistics and Operations Research Transactions*, 38(2):285–302, 2014. [p109, 110, 111]

E. J. Beh and R. Lombardo. *Correspondence Analysis: Theory, Practice and New Strategies*. John Wiley & Sons, 2014. [p109]

J. Benzécri. *Analyse des Données*. Dunod, 1973. [p109]

U. Böckenholt and Y. Takane. Linear constraints in correspondence analysis. In M. Greenacre and J. Blasius, editors, *Correspondence Analysis in the Social Sciences*. Academic Press, 1994. [p111]

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. [p112]

B. Escofier and J. Pagès. *Analyses factorielles simples et multiples*. Dunod, 1988. [p111, 112]

M. Greenacre. *Theory and Applications of Correspondence Analysis*. Academic Press, 1984. [p112]

M. Greenacre. *Correspondence Analysis in Practice*. CRC Press, 2007. [p109]

F. Husson, S. Lê, and J. Pagès. *Exploratory Multivariate Analysis by Example Using R*. Chapman & Hall / CRC Press, 2010. [p109]

S. Lê, J. Josse, and F. Husson. FactoMineR: An R package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18, 2008. [p109]

L. Lebart, A. Salem, and L. Berry. *Exploring Textual Data*. Kluwer, 1998. [p109, 110, 112]

F. Murtagh. *Correspondence Analysis and Data Coding with Java and R*. Chapman & Hall / CRC Press, 2005. [p109, 110]

Belchin Kostov
Transverse group for research in primary care, IDIBAPS
Mejia Lequerica, s / n.
08028 Barcelona
Spain
badriyan@clinic.ub.es

Mónica Bécue-Bertaut
Department of Statistics and Operational Research
Universitat Politècnica de Catalunya
North Campus – C5
Jordi Girona 1–3
08034 Barcelona
Spain
monica.becue@upc.edu

François Husson
Agrocampus Rennes
65 rue de Saint-Brieuc
35042 Rennes
France
husson@agrocampus-ouest.fr

Implementing Persistent $O(1)$ Stacks and Queues in R

by Shawn T. O'Neil

Abstract True to their functional roots, most R functions are side-effect-free, and users expect datatypes to be persistent. However, these semantics complicate the creation of efficient and dynamic data structures. Here, we describe the implementation of stack and queue data structures satisfying these conditions in R, available in the CRAN package **rstackdeque**. Guided by important work in purely functional languages, we look at both partially- and fully-persistent versions of queues, comparing their performance characteristics. Finally, we illustrate the usefulness of such dynamic structures with examples of generating and solving mazes.

Introduction

Dynamic data structures such as trees, hash tables, heaps, stacks and queues are fundamental to a wide variety of algorithmic strategies, especially those that are recursive in nature (Cormen et al., 2001). While trees and hash tables are often used for simple name/value lookups—supported well by environments and named lists—stacks (first in, last out) and queues (first in, first out) are used precisely for their efficient insertion and removal operations. While these structures can be implemented via pre-allocated vectors and lists when combined with additional start/end index information, this requires foreknowledge of their maximum size (or a dynamic resizing strategy) as well as careful bookkeeping.

Most implementations of stacks and queues are mutable; they support `push()` and `pop()` operations that have the *side effects* of modifying or mutating the data structure in place for insertion and removal. Although R does support the development of mutating data structures, most common structures do not mutate (without, at least, an explicit assignment) and developers are encouraged to follow this paradigm and create side-effect-free functions. In addition to providing a consistent programming environment, this admonition also facilitates the creation of easily parallelized code (Böhringer, 2013).

More formally, we say that a data structure is "persistent," if, after an insertion or removal the original version is still accessible. To the programmer, it appears as if a modified copy of the structure was returned. Such persistence provides the side-effect-free nature of accessor functions expected by users of functional languages. A structure is "fully persistent" if previous versions support further insertions and removals, and "partially persistent" if previous versions may only be inspected.

Fortunately, the implementation of persistent data structures in purely functional languages is well studied and continues as an active area of research (Chuang and Goldberg, 1993; Kaplan, 1995; Okasaki, 1999). For languages that support certain features there now exist persistent balanced binary trees, queues, stacks, heaps (Okasaki, 1999), union-find structures (Conchon and Filliâtre, 2007), and many others as fast or nearly as fast as their mutating counterparts. The solutions are often deeply recursive and take advantage of tail-call optimization and delayed-evaluation provided by languages such as Haskell and ML.

As a quick review, a stack is a first-in, last-out structure: elements are removed in the opposite order they are inserted. Usually, we think of insertions and removals occurring at the "top" of a stack. Stacks are used in a number of programming paradigms, especially depth-first searches of state spaces for optimization and search problems. Queues are first-in, first-out: elements are removed in the same order in which they are inserted, and we usually imagine inserting at the "back" and removing from the "front" of a queue. Deques, or double-ended queues, additionally support insertion and removal from both ends. Queues and deques are used for breadth-first searches of state spaces and also find use in simulation contexts (e.g. Grether et al. 2012). All three structures generally also support "peeking," or inspection of elements that are ready for removal.

Ideally, insertion, removal, and peeking on stacks, queues, and deques would be nearly instantaneous no matter their size ($O(1)$ worst-case time). Mutable versions of these structures are easy to implement and provide this speed, but persistence complicates matters. In practice, *amortized* $O(1)$ operations are an excellent compromise; a few individual operations may be slow ($O(n)$ or $O(\log n)$), but n insertions and/or removals are guaranteed to run in $O(n)$ worst-case time overall, for an $O(1)$ time-per-operation on average. Here, we would like to additionally use a new term: "slowly persistent," to describe a structure that is fast when used partially persistently, but may be slower when used in a fully-persistent context.

In the following sections, we describe the implementation of $O(1)$ worst-case, fully persistent stacks in R (which we call 'rstack's), slowly persistent $O(1)$ amortized deques ('rdeque's), and finally $O(1)$ worst-case fully persistent queues ('rpqueue's) as described by Okasaki (1995).

Fully persistent stacks

For languages that disallow mutability entirely, even stack implementations can be moderately complex. In R, however, we can take advantage of mutability via environments and simply provide side-effect-free interface functions. We implement stacks as unordered linked-lists of environments, each with a data element and a reference to the next node. Stacks themselves are S3 objects with interface methods and access to the head stack node as well as length information stored separately.

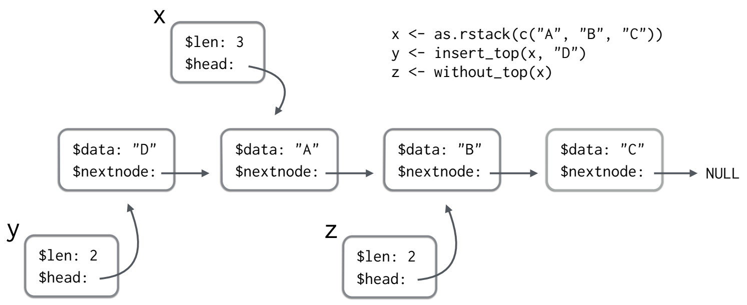


Figure 1: Illustration of a fully persistent stack as a linked-list.

To emphasize the persistent nature of the structure, rather than implement the traditional `push()` and `pop()` methods, we instead use `insert_top()` (returning a new 'rstack' object with a new top element), `without_top()` (returning a new 'rstack' object without the top element) and `peek_top()` (returning the data element of the top of the stack, or an error if the stack is empty–though we also implement `empty()` for explicit tests of emptyness). As illustrated in Figure 1, insertion and removal are implemented as creation of a new stack with head element referencing the appropriate node, for $O(1)$ worst-case time for all operations as well as full persistence.

Slowly persistent queues and deques

The simplest way to implement persistent queues has long been known (Burton, 1982), and simply keeps two stacks: a "left" that holds the elements at the front of the queue ready for removal, and a "right" holding the elements at the back of the queue in reverse order (where new elements are added).

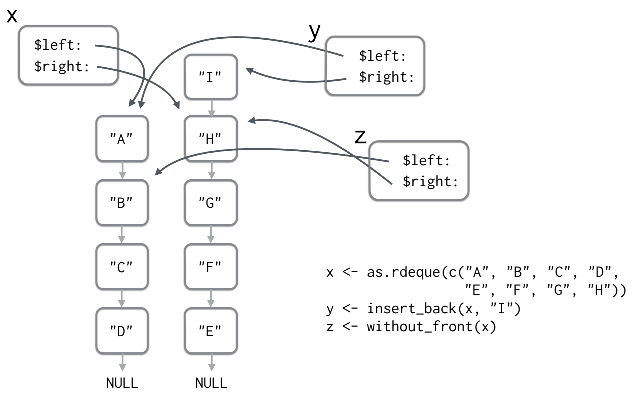


Figure 2: Illustration of a slowly persistent queue as a pair of stacks.

Of course, we need to be able to handle removing from a non-empty queue when the `left` stack is empty. In this case we need to first reverse the `right` stack and put it in the `left` position. This is an $O(n)$ operation, but one whose average expense is inversely proportional to its frequency in any series of insertions/deletions, for $O(1)$ amortized time provided the queue is used only partially persistently (see Okasaki 1999 for a rigorous proof and thorough discussion of amortization).

In actuality, we implement this structure as a double ended queue for R (an 'rdeque') which allows for fast access to both ends. When one side becomes nearly empty, returned structures are first decomposed to a list and recomposed with balanced sides, again providing $O(1)$ amortized time when used partially persistently. They are still available for persistent use, but in this case amortized $O(1)$ time cannot be guaranteed. For example, if `y <- without_front(x)` causes an $O(n)$ rebalance, then so will all similar calls such as `z <- without_front(x)`. In practice, we expect usage patterns that cause significant slowdown to be rare.

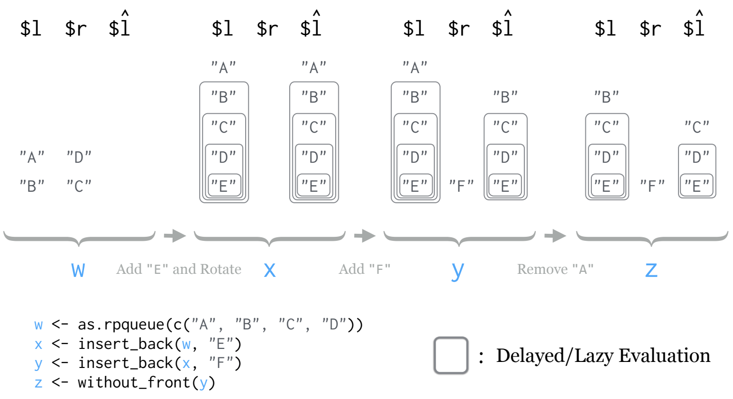
Fast fully persistent queues

True fully persistent, $O(1)$ worst-case queues are also known, but these solutions rely heavily on recursively defined operations and delayed evaluation (Okasaki, 1995). The building block for many functional data structures is the lazy list, a structure whose first element is immediately accessible, but whose "tail" (the rest of the elements) are evaluated in a delayed fashion. Interestingly, this allows for the recursive definition of infinite lists such as the Fibonacci sequence, where successive elements are syntactically defined but only evaluated when accessed (Paulson, 1996).

We implement these persistent queues as 'rpqueue''s, which require three lazy lists: `l`, `r`, and $\hat{\texttt{l}}$. Items are inserted into the back of the queue by addition to the top of the `r` list, and removed from the front by removal from the front of the `l` list. We implement these lazy lists as 'rstack''s (described above) where nodes' `nextnode` elements are assigned with `delayedAssign()` and memoized on first evaluation.

The queue maintains the invariant that the `l` stack is at least as long as the `r` stack–when this invariant would be violated (either by an insertion to `r` or a removal from `l`), the `r` list is rotated and appended to the tail of `l`; this operation is delayed for $O(1)$ insertions and removals. Simultaneously, the $\hat{\texttt{l}}$ list is set to contain the same data as `l` on a rotation. For each subsequent insertion or removal (which don't cause a rotation and are also constant time), an element is also removed from $\hat{\texttt{l}}$. This ensures that future removals are constant time as previously delayed computations are forced even on insertions. (The rotation is incremental, and so all tails of `l` and $\hat{\texttt{l}}$ are delayed.) With these operations, the $\hat{\texttt{l}}$ queue is empty exactly when the invariant is violated causing another delayed rotation. For further details, see Okasaki (1995).

An example operation is shown in Figure 3, where an initial queue of `"A"`, `"B"`, `"C"`, `"D"` receives an insertion of `"E"`, causing a rotation. Next, an insertion of `"F"` does not cause a rotation but does cause a removal from $\hat{\texttt{l}}$, pre-evaluating the answer required for the upcoming removal of `"A"`. When the removal of `"A"` does occur, the operation is $O(1)$ and another element is removed from $\hat{\texttt{l}}$ anticipating future removals from `l`.



```
w <- as.rpqueue(c("A", "B", "C", "D"))
x <- insert_back(w, "E")
y <- insert_back(x, "F")
z <- without_front(y)
```

Figure 3: Example of $O(1)$ worst-case 'rpqueue' operation, based on three stacks (`l`, `r`, and $\hat{\texttt{l}}$) where the "tails" of stacks may be assigned with `delayedAssign()` in the incremental rotation and are memoized upon first evaluation. Items are inserted onto `r` and removed from `l`. The $\hat{\texttt{l}}$ stack is reduced by one on each insertion or removal, ensuring that future versions of `l` have been pre-evaluated for constant time removals.

Because lazy lists are potentially deeply nested, we can't fully evaluate them recursively due to interpreter call-stack limitations. In a similar vein, an important property of these purely-functional queues from an implementation perspective is that they are indeed $O(1)$ worst-case. Thus, the evaluation cascade that might result from delayed assignment is never deeper than $O(1)$ for any insertion or removal, and we needn't worry about call-stack overflows.

Features

The three data structures we've described, '`rstack`', '`rdeque`', and '`rpqueue`' are all available in the R package, **rstackdeque**, on CRAN. All provide an `as.list` convenience function that collates all elements into a pre-allocated list of the correct size with the top/front elements being placed in the first index of the list. Similarly, `as.rstack()`, `as.rdeque()`, and `as.rpqueue()` convert their inputs into a stack, deque, or fully persistent queue, after converting to an intermediary list representation with `as.list()`. These bulk operations are somewhat slower for '`rpqueue`'s because of their delayed nature and the need to process modifications incrementally, in addition to the supporting structures needed for their implementation. All three structures also support modification by direct assignment to their tops or ends, as in `peek_top(x) <- "A"` for stacks, `peek_front(x) <- "A"` for queues, and additionally `peek_back(x) <- "A"` for deques.

While these examples have illustrated storing simple character vectors in stacks and queues, any R data type may be stored efficiently. One of the more common errors for new R programmers with experience in non-functional languages is the attempt to use a loop to dynamically grow a data structure like a dataframe. As experienced R programmers know, the persistent nature of dataframes but lack of efficient growth capabilities may result in excessive copying and $O(n^2)$ behavior. By contrast, with $O(1)$ insertion provided by stacks and queues, the programmer can easily and quickly insert single-row dataframes (or lists representing rows) and later extract and collate them into a full dataframe in $O(n)$ total time. The structures described here thus also come with a convenience function `as.data.frame()` that efficiently collates elements, so long as they all have the same `length()` and if any are named the names do not conflict. (This is accomplished with a call to `rbind` via `do.call` on a list-representation of the structure.) We emphasize that this feature is not meant as a replacement for the functional solutions provided in base R like `lapply()` or excellent split-apply-combine strategies such as in the **dplyr** package (Wickham and Francois, 2015). Rather, we hope that stacks and queues will provide R programmers with highly dynamic data structures that are useful for certain applications while fitting well with the existing R ecosystem.

Performance comparison

We briefly wish to compare the performance characteristics of these three data structure implementations. Although certainly much slower than less dynamic pre-allocated-list solutions, persistent stacks, deques, and queues should be efficient enough to be used naturally when they are most appropriate. In addition, performance comparisons between the two implementations of queues described above—partially persistent and fully persistent based on lazy evaluation—would be the first to our knowledge.

Using the **microbenchmark** package (Mersmann, 2014), we tested the time taken to 1) insert n items, 2) remove n items, and 3) perform a mix of n insertions and deletions, where a deletion followed every two insertions. Results can be found in Figure 4.

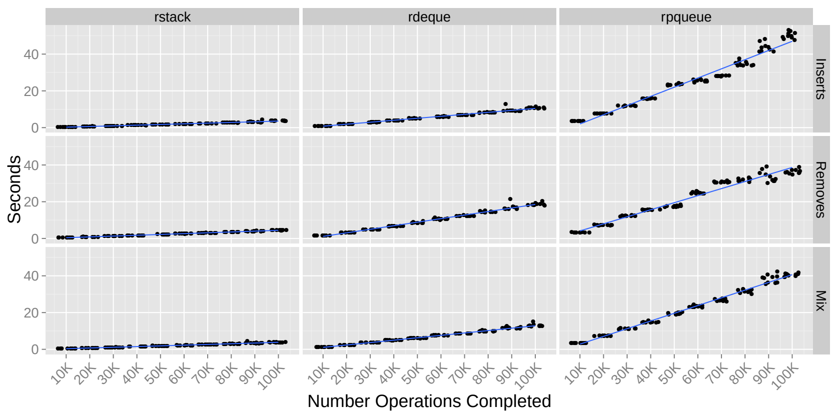


Figure 4: Benchmark timings for n insertions, n removals, and a mix of n insertions and removals for '`rstack`'s, '`rdeque`'s, and '`rpqueue`'s. Each point represents one trial, with jitter and linear model fits added for clarity. The close fit of the linear models to the points reinforces the $O(1)$ amortized time for all three structures in these tests.

Generally, '`rdeque`' structures are about twice as slow as '`rstack`'s on insert and four times slower on remove, reflecting the additional environment references on insert and occasional rebalancing needed for removals. The 'rpqueue's are slower still (by a constant factor per element) owing to their complex insertion and removal routines.

Because of the supporting structures needed for 'rdeque's and 'rpqueue's, we also looked at how many environments were created by *n* insertions as reported by `memory.profile()` immediately after *n* insertions. (Before each set of insertions, the structures were removed with `rm()` and the garbage collector was run with `gc()`.) Results are shown in Figure 5. As expected, the partially persistent stacks and deques use (asymptotically) one environment per data element. The number of environments utilized by the 'rpqueue's varies depending on the internal evaluation state of the structure, with at some points several environments supporting each data element. Note that because the insertion time is $O(1)$ worst-case, the total number of environments must also be $O(n)$ after *n* insertions.

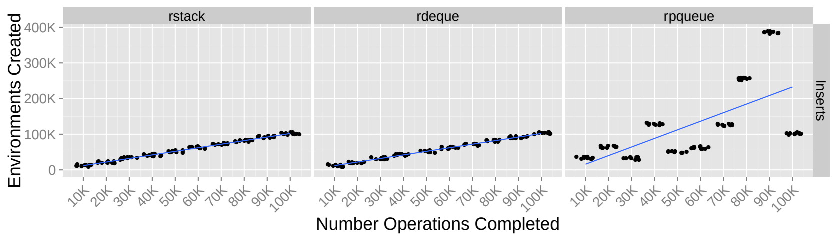


Figure 5: Benchmark environment counts after *n* insertions for 'rstack's, 'rdeque's, and 'rpqueue's. Each point represents one trial, with jitter and linear model fits added for clarity.

Example usage

To highlight the usefulness of stacks and queues as well as give more in-depth examples, we consider a whimsical application that makes heavy use of these structures: designing and solving mazes.

There are a variety of ways to design a maze; one of the more elegant is known as "recursive backtracking." This strategy builds corridors by extending them from the start in random directions, inserting the cells as they are visited onto the top of a stack. When the corridor would intersect itself, the path is traced backward (removing from the stack) and another corridor is opened up. A small ASCII maze built with this method is shown in Figure 6.

```
#######################################################
#######################################################
##E#           #         #                       #   ##
## # ######### # ##### # ##### ################# # # ##
## #   # #     # #     #       #   #         # # # # ##
## ### # # ####### ############# ### ##### # # # # # ##
##     # #       # #     #   #       #     #   # # # ##
######## ####### # # ### # # ### ##### ####### # ### ##
##       #       #     #   #   #   #   #   #   #   # ##
## ####### ################### ##### ### ### ##### # ##
##        #   #       #       #     #   #   #     # ##
## ######### # # # ### # ####### ##### ### ### ##### ##
##       #   # # # #   # #   #   #     #     # #   # ##
######## ### # # # # ### # ### # ##### # ##### # # # ##
## #   #     # # # #     # #   # #   #   # #   # #   ##
## # # ####### ### ####### # ### # # # ### # ### ### ##
##   #       #   # #         # #   # #     # #   # # ##
## ####### ##### # # ######### ##### ##### # # ### # ##
## #     # #   # # #         #   #   #   # # #   # # ##
## # ### # # # # # ######### ### # ### # ### ### # # ##
##   #   #   #     #             #     #         #  S##
#######################################################
#######################################################
```

Figure 6: Example ASCII maze generated via recursive backtracking supported by a stack.

The R code to generate an ASCII maze like this is somewhat tedious so we won't reproduce it here except in pseudocode (though see Section Availability):

```
## Maze generation pseudocode
V <- rstack()
set the current location L to the start of the maze at S
while(any cell is unvisited) {
  find a random unvisited neighbor N of L
  if(N is not NULL) {
    V <- insert_top(V, L)
    build a corridor between N and L
    L <- N
    mark L visited
  else if(the stack is not empty)
    # pull the next location to try from stack
    L <- peek_top(V)
```

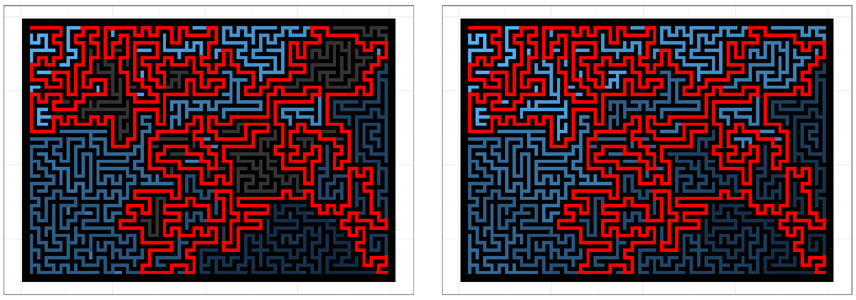


Figure 7: Maze solved with depth-first search supported by a stack (left) and breadth-first search supported by a queue (right); lighter colored cells were visited later by the search and the solution (from the start in the lower right to the end in the upper left) is shown in red.

```
      V <- without_top(V)
    else
      set L to a random unvisited cell in the maze
```

Solving mazes of this type using depth-first search is relatively simple with the help of a stack. The start location is inserted into the top of the stack, and so long as peeking at the top of the stack doesn't reveal the end location, neighboring unvisited locations are inserted. If there are no unvisited locations nearby, indicating a dead end, the top of the stack is removed instead until the search can continue. (This function calls a helper function, `random_unvisited_neighbor`, see Section Availability for details.)

```
  solve_ascii_maze_dfs <- function(maze) {
    ## find the start and end of the maze
    end <- data.frame(which(maze == "E", arr.ind = TRUE))
    start <- data.frame(which(maze == "S", arr.ind = TRUE))

    maze[maze == " " | maze == "E" | maze == "S"] <- "." ## mark corridors unvisited

    loc <- start ## initialize the solution stack and start location
    path <- rstack()
    path_history <- rstack()
    path <- insert_top(path, loc)

    step <- 1
    while(any(peek_top(path) != end)) { ## while we're not out of the maze
      loc <- peek_top(path)

      ## if the current location is unvisited, mark it visited with the current
      ## timestep
      if(maze[loc$row, loc$col] == ".") {maze[loc$row, loc$col] <- step}

      ## grab a random unvisited neighbor; if there is one push it on the current
      ## path
      nextloc <- random_unvisited_neighbor(maze, peek_top(path), dist = 1)
      if(!is.null(nextloc)) {
        path <- insert_top(path, nextloc)
      } else {
        # otherwise backtrack and try again
        path <- without_top(path)
      }
      step <- step + 1
      path_history <- insert_top(path_history, path)
    }

    end <- peek_top(path) ## mark solution by negating visit numbers
    maze[end$row, end$col] <- step
    while(!empty(path)) {
```

```
    loc <- peek_top(path)
    path <- without_top(path)
    maze[loc$row, loc$col] <- -1 * as.numeric(maze[loc$row, loc$col])
  }
  return(list(maze, path_history))
}
```

In this implementation we mark each cell as visited with the timestep in which it is visited, and in the end we reverse those numbers for the solution by tracing back through the solution path stored in the stack. An auxiliary function plots the solved ASCII maze, including this solution path, resulting in Figure 7 (left) for a larger maze.

A breadth-first search replaces the stack with a queue, so that new cells are visited in order of their distance from the start—cells further from the start are added to the back of the queue for later visiting and cells are marked visited from the front of the queue. One small complication is the need to keep track of where cells are sampled from relative to their neighbors so that we can trace back the eventual solution. We do this with a simple stack that keeps track of this to/from information and use a while-loop for the traceback.

```
solve_ascii_maze_bfs <- function(maze) {
  ## find the start and end of the maze
  end <- data.frame(which(maze == "E", arr.ind = TRUE))
  start <- data.frame(which(maze == "S", arr.ind = TRUE))

  maze[maze == " " | maze == "E" | maze == "S"] <- "." ## mark corridors unvisited

  loc <- start ## initialize the solution stack and start location
  visits <- rdeque()
  visits_history <- rstack()
  visits <- insert_back(visits, loc)

  ## keep a stack to remember where each visit came from in the BFS
  camefrom <- rstack()

  step <- 1
  while(!empty(visits)) { ## while there are still cells we can visit
    loc <- peek_front(visits)
    visits <- without_front(visits)

    neighbors <- random_unvisited_neighbor(maze, loc, dist = 1, all = TRUE)
    for(neighbor in neighbors) {
      camefrom <- insert_top(camefrom, list(from = loc, to = neighbor))

      ## push neighbors on the queue and mark them visited with the timestep
      visits <- insert_back(visits, neighbor)
      maze[neighbor$row, neighbor$col] <- step
      step <- step + 1
      visits_history <- insert_top(visits_history, visits)
    }
  }

  loc <- end ## set loc to the end and track the camefrom path back to find the solution
  while(any(loc != start)) {
    maze[loc$row, loc$col] <- as.numeric(maze[loc$row, loc$col]) * -1
    pathpart <- peek_top(camefrom)
    while(any(pathpart$to != loc)) {
      camefrom <- without_top(camefrom)
      pathpart <- peek_top(camefrom)
    }
    loc <- pathpart$from
  }
  maze[loc$row, loc$col] <- as.numeric(maze[loc$row, loc$col]) * -1
  return(list(maze, visits_history))
}
```

This solution will run longer (as it will visit all of the maze rather than stop the first time the end is

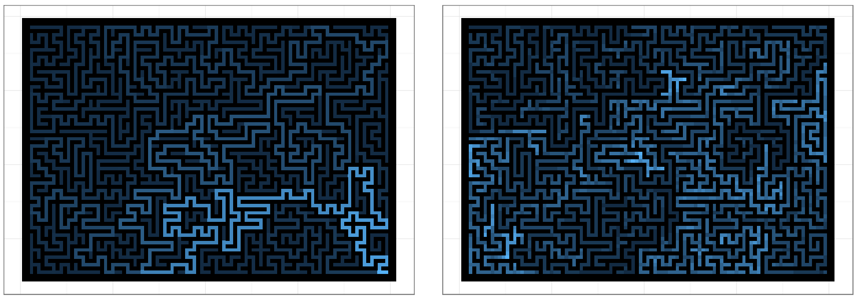


Figure 8: Mazes illustrating the length of time each cell spent waiting in the search structure, with lighter colors indicating longer periods in the 'rstack' for the depth-first search (left) or 'rdeque' for the breadth-first search (right).

found), but is guaranteed to find the shortest path from the start to the end. In this case both solutions are the same, as shown in Figure 7 (right). (In fact, mazes generated by recursive backtracking are trees, so there is only one simple solution.)

These algorithmic examples don't make explicit use of the persistence provided by 'rstack's and 'rdeque's. Some persistent data structures are an integral part of the methods they are used in; for example persistent binary trees are used in the planar point location problem (Sarnak and Tarjan, 1986), and many other applications of persistence are found in the area of computational geometry in general (Kaplan, 1995). Persistence also supports multi-threaded parallelization, since different threads needn't worry about the mutability of their parameters (Chuang and Goldberg, 1993).

More simply for our examples, because the structures are persistent we can save the state of the search over time (in `path_history` and `visits_history` in the functions above). These histories can later be analyzed, for example to illustrate how many steps each cell spent waiting in the stack or deque (Figure 8).

Availability

The stack, deque, and queue structures described here are available as an R package called **rstackdeque** available on CRAN. The maze examples and benchmarking code may be found on GitHub at `https://github.com/oneilsh/rstackdeque_examples`. Persistent structures such as these also support illustrating the operation of dynamic algorithms with the powerful plotting capabilities of **ggplot2** (Wickham, 2009) and similar packages. As an example, the `mazes.mp4` video in the `rstackdeque_examples` GitHub repository illustrates the creation and solution of a maze through time.

Summary

We've described the implementation of dynamic stack and queue data structures, guided by research in purely-functional languages so that they fit well into the R ecosystem via persistence. Rather than serve as a replacement for standard containers like lists and dataframes, stacks and queues (along with hashes as provided by the **hash** package (Brown, 2013)) enable a wide range of staple algorithmic techniques that until now have been more naturally implemented in languages like Python, Java, and C.

Further work might consider implementing additional persistent data structures, such as red-black trees or heaps. Because persistent versions of these are often $O(\log n)$ amortized insertion and removal, it may be beneficial to implement them via **Rcpp** and functional approaches for C++ (McNamara and Smaragdakis, 2000; Eddelbuettel and François, 2011; Eddelbuettel, 2013).

Acknowledgements

Thanks to Zhian Kamvar and Charlotte Wickham for comments, and the Center for Genome Research and Biocomputing at Oregon State University for support. Special thanks to reviewers, whose suggestions improved clarity and motivation for this article as well as contributed to the usability of the **rstackdeque** package.

Bibliography

S. Böhringer. Dynamic parallelization of R functions. *The R Journal*, 5(2):88–96, 2013. [p118]

C. Brown. *hash: Full Feature Implementation of Hash/Associated Arrays/Dictionaries*, 2013. URL `http://CRAN.R-project.org/package=hash`. R package version 2.2.6. [p125]

F. W. Burton. An efficient functional implementation of FIFO queues. *Information Processing Letters*, 14 (5):205–206, 1982. [p119]

T.-R. Chuang and B. Goldberg. Real-time deques, multihead Turing machines, and purely functional programming. In *Proceedings of the Conference on Functional Programming Languages and Computer Architecture*, pages 289–298, 1993. [p118, 125]

S. Conchon and J.-C. Filliâtre. A persistent union-find data structure. In *Proceedings of the 2007 Workshop on ML*, pages 37–46. ACM, 2007. [p118]

T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001. [p118]

D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. [p125]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. [p125]

D. Grether, A. Neumann, and K. Nagel. Simulation of urban traffic control: A queue model approach. *Procedia Computer Science*, 10:808–814, 2012. [p118]

H. Kaplan. Persistent data structures. In *Handbook on Data Structures and Applications*, pages 241–246. CRC Press, 1995. [p118, 125]

B. McNamara and Y. Smaragdakis. Functional programming in C++. *ACM SIGPLAN Notices*, 35(9): 118–129, 2000. [p125]

O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2014. URL `http://CRAN.R-project.org/package=microbenchmark`. R package version 1.4-2. [p121]

C. Okasaki. Simple and efficient purely functional queues and deques. *Journal of Functional Programming*, 5(04):583–592, 1995. [p119, 120]

C. Okasaki. *Purely Functional Data Structures*. Cambridge University Press, 1999. [p118, 119]

L. Paulson. *ML for the Working Programmer*. Cambridge University Press, 1996. [p120]

N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29(7):669–679, 1986. [p125]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, New York, 2009. URL `http://had.co.nz/ggplot2/book`. [p104, 125]

H. Wickham and R. Francois. *dplyr: A Grammar of Data Manipulation*, 2015. URL `http://CRAN.R-project.org/package=dplyr`. R package version 0.4.2. [p121]

Shawn T. O'Neil
Center for Genome Research and Biocomputing
Oregon State University
3021 ALS, Corvallis, OR 97333
United States of America
`shawn.oneil@cgrb.oregonstate.edu`

R as an Environment for Reproducible Analysis of DNA Amplification Experiments

by Stefan Rödiger, Michał Burdukiewicz, Konstantin Blagodatskikh, Michael Jahn and Peter Schierack

Abstract There is an ever-increasing number of applications, which use quantitative PCR (qPCR) or digital PCR (dPCR) to elicit fundamentals of biological processes. Moreover, quantitative isothermal amplification (qIA) methods have become more prominent in life sciences and point-of-care-diagnostics. Additionally, the analysis of melting data is essential during many experiments. Several software packages have been developed for the analysis of such datasets. In most cases, the software is either distributed as closed source software or as monolithic block with little freedom to perform highly customized analysis procedures. We argue, among others, that R is an excellent foundation for reproducible and transparent data analysis in a highly customizable cross-platform environment. However, for novices it is often challenging to master R or learn capabilities of the vast number of packages available. In the paper, we describe exemplary workflows for the analysis of qPCR, qIA or dPCR experiments including the analysis of melting curve data. Our analysis relies entirely on R packages available from public repositories. Additionally, we provide information related to standardized and reproducible research.

Introduction

Quantitative polymerase chain reaction (qPCR) is the method of choice when a precise quantification of minute DNA traces is required. Applications include the detection and quantification of pathogens or gene expression analysis (Pabinger et al., 2014). Only few bioanalytical methods had such a significant impact on the progress of life sciences and medical sciences as the qPCR (Huggett et al., 2015b). Numerous commercial and experimental monitoring platforms have been developed in the past years. This includes standard plate cyclers, capillary cyclers, microfluidic platforms and related technologies (Rödiger et al., 2013b; Viturro et al., 2014; Rödiger et al., 2014; Khodakov and Ellis, 2014; Wu et al., 2014).

In the past decades several isothermal amplification technologies emerged, such as helicase dependent amplification (HDA). Isothermal amplification methods were readily combined with real-time monitoring technologies (qIA) or digital PCR and are used in various fields like diagnostics and point-of-care-testing (Selck et al., 2013; Rödiger et al., 2014; Nixon et al., 2014).

Digital PCR (dPCR) is a novel approach for detection and quantification of nucleic acids and can be seen as a next generation method (Huggett et al., 2015b). The dPCR technology breaks fundamentally with the previous concept of nucleic acid quantification. In contrast to traditional qPCR measures dPCR absolute nucleic acids amounts. This is possible after ‘clonal DNA amplification’ in thousands of small separated partitions (e.g., droplets, nano chambers; Huggett et al. 2013; Milbury et al. 2014; Morley 2014). Partitions with no nucleic acid remain negative and the others turn positive (e.g., Figure 1). Selected technologies (e.g., OpenArray®real-time PCR system) monitor amplification reactions in the chambers (‘partitions’) in real-time. After that, all quantification cycle (Cq) values are calculated from the amplification curves and converted into discrete events of positive and negative chambers. Finally, the absolute quantification of nucleic acids is done using Poisson statistics (Milbury et al., 2014; Morley, 2014). A representative workflow using the vendor software for the analysis of a droplet dPCR experiment has been described by Milbury et al. (2014). Recently, we have published the **dpcR** package (Burdukiewicz et al., 2015) at CRAN, which is the first open source R software package for the analysis of dPCR experiments (see the **dpcR** package manual for details).

The complexity of hardware, wetware and software requires expertise to master a workflow. This comprises standards for experiment design, generation and analysis of data, multiple hypothesis testing, interpretation, reporting and storage of results (Huggett et al., 2014; Castro-Conde and de Uña-Álvarez, 2014). Scientific misconduct and fraud have shaken the scientific community on several occasions (Bustin, 2014). In particular, the scientific community works hard to uncover pitfalls of qPCR experiments. This led to the development of peer-reviewed quantification cycle (Cq) analysis algorithms (Ruijter et al., 2013), fully characterized qPCR chemistries (Ruijter et al., 2014) and guidelines for a proper conduct of qPCR experiments as implemented in the MIQE guidelines (minimum information for publication of quantitative real-time PCR experiments; Huggett et al. 2013; Bustin 2014). We share the philosophy of the MIQE guidelines to increase the experimental transparency for better experimental practice and reliable interpretation of results and encourage the

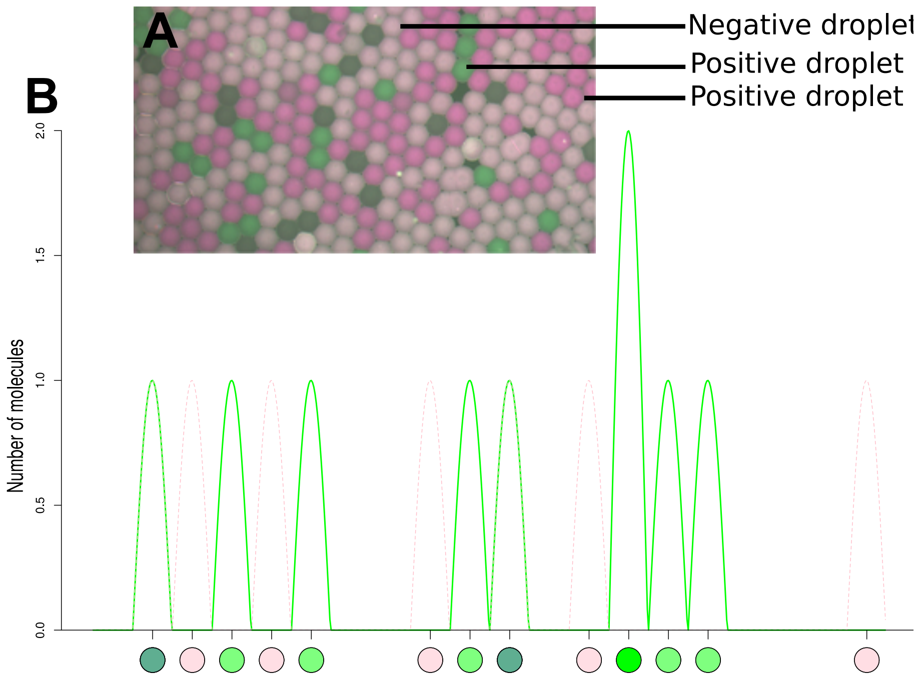


Figure 1: Scheme of a digital PCR experiment. **(A)** A droplet digital PCR reaction mix was formed in a BioRad QX100 Droplet Digital PCR System. Droplets (circa 100 μm in diameter) were subjected to custom made slide chambers for the detection and analysis in fully automatized imaging VideoScan platform (Rödiger et al., 2013b). **(B)** Subsequently the samples can be digitalized by counting the number of positive and total number of droplets. The plot was generated with the `sim_ddpcr` function from the **dpcR** package.

use of open data exchange formats like the XML-based real-time PCR data markup language (RDML; Lefever et al. 2009).

In case of closed source software the analysis usually happens in a black box fashion tied to a specific platform. We agree that closed frameworks are not necessarily a bad thing, but should be avoided if possible (Rödiger et al., 2013a; Spiess et al., 2015). Studies by McCullough and Heiser (2008); Almiron et al. (2010); Durán et al. (2014) exemplified where the black box approach might fail in science. The same holds true for the open source software since software bugs are independent of a development model. However, at least open source gives the possibility to track and eliminate errors by an individual entity. Black-box systems often force the user to process the data by suboptimal analysis algorithms (Ruijter et al., 2013). Moreover, the data usually cannot be accessed between the steps of the analysis, which restrains quality control. Visualization options are usually limited by the software vendor preferences and do not attain publication quality. In addition to this, users who have no access to the commercial software are barred. Aside from closed source software, data analysis is often performed in spreadsheets. However, this data processing approach is not advisable for research purposes. Most spreadsheets lack (or do not use) tools to validate the input, to debug implemented procedures and to automatize the workflow. These traits make them prone to errors and not well suited for complicated tasks (McCullough and Heiser, 2008; Burns, 2014).

For several reasons, R is one of the most popular tools in bioinformatics and is known as an early adopter of emerging technologies (Pabinger et al., 2014). R provides packages to build highly customized workflows, covering: data read-in, data pre-processing, analysis, post-processing, visualization (Murrell, 2012, 2015) and storage. As recently briefly reviewed in Pabinger et al. (2014), numerous R packages have been developed for the analysis of qPCR, dPCR, qIA and melting curve analysis experiments, including: **kulife** (Ekstrom et al., 2013), **MCMC.qpcr** (Matz, 2015), **qPCR.CT** (Pan et al., 2012), **DivMelt** (Swan, 2013), **qpcR** (Spiess, 2014), **dpcR**, **chipPCR** (Roediger and Burdukiewicz, 2014), **MBmca** (Roediger, 2015), **RDML** (Blagodatskikh et al., 2015), **nondetects** (McCall1 et al., 2014), **qpcrNorm** (Mar, 2009), **HTqPCR** (Dvinge and Bertone, 2009), **SLqPCR** (Kohl, 2007), **ddCt** (Zhang et al., 2015), **EasyqpcR** (Pape, 2012), **unifiedWMWqPCR** (Neve et al., 2014; Neve and Meys, 2014), **ReadqPCR** and **NormqPCR** (Perkins et al., 2012) All packages are either available from CRAN or Bioconductor (Gentleman et al., 2004) and can be freely combined in a plugin-like architecture.

R is an open, operating system-independent platform for a broad spectrum of calculation options. Particularly, the visualization of experiments is one of R's pinnacles. R enables the users to create an

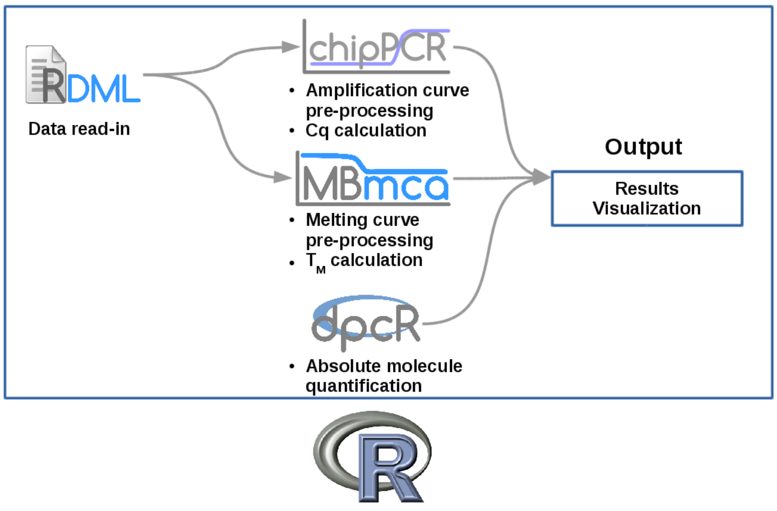

RDML
Data read-in
chipPCR
• Amplification curve pre-processing
• Cq calculation
MBmca
• Melting curve pre-processing
• T_M calculation
dpcR
• Absolute molecule quantification
Output
Results Visualization

Figure 2: An exemplary workflow for quantitative PCR, digital PCR, quantitative isothermal amplification and melting curve analysis experiments in R. The R software environment provides core functionality for statistical computing and graphics. In our scenario we used the **RDML** package to read-in data in standardized format. However, any format supported by R can be used. Further, processing of amplification curve data was performed with the **chipPCR** package and melting curve data were analysed with the **MBmca** package. The **dpcR** package can be embedded in the analysis of digital PCR experiments. Cq, quantification cycle; T_M, melting temperature.

efficient manipulation, restructuring and reshaping of data to make them readily-available for further processing. This is of the particular importance to the human–machine interface (Oh, 2014). Intrinsic properties of R such as naming conventions (Bååth, 2012) and class systems (e.g., **S3**, **S4**, reference classes and **R6**) vary considerable, depending on the package developer preferences. However, due to the open source approach, there is the common ground to track numerical errors. R offers various methods for a standardized data import/export and exchange. Workflows can embed structured models (Zeller et al., 2009), open data exchange formats (e.g., RDML), binary formats (Michna and Woods, 2013) or tools provided by the R workspace (R Core Team, 2015). The NetCDF binary format, available from the **RNetCDF** package (Michna, 2015), has advantages over some other binary formats (e.g., the RData format), since arbitrary array data sections of massive datasets can be processed efficiently (Michna and Woods, 2013). This might be useful for large data sets as present in high-throughput PCR experiments or dPCR experiments with large partition numbers. The R environment offers several datasets, which can be used for testing of algorithms. Therefore, we, among others, argue that R is suitable for reproducible research (Gentleman et al., 2004; Murrell, 2012; Gandrud, 2013; Hofmann et al., 2013; Ooms, 2013; Kuhn, 2015; Leeper, 2014; Liu and Pounds, 2014).

The aim of this paper is to show case studies for qPCR, dPCR, qIA and melting curve analysis experiments. Our workflow effectively follows the principle illustrated in Figure 2. We intend to aggregate functionalities dispersed between various packages and offer a fast insight in the analysis of nucleic acid experiments with R. In particular, we describe how to:

- read-in data from a standardized file format,
- pre-process the amplification curve data,
- calculate specific parameters from the amplification curve data,
- calculate the melting temperature,
- and report the data.

Setting-up a working environment

We recommend performing the scripting in a dedicated integrated development environment (IDE) and graphical user interface (GUI) such as RKWard (Rödiger et al., 2012), RStudio (RStudio, 2014; Gandrud, 2013) or other technologies (Valero-Mora and Ledesma, 2012). Benefits of IDEs with GUI include syntax-highlighting, auto completion and function references for rapid prototyping of workflows. Typically, the analysis will start with data from a commercial platform. Most platforms have an option to export a CSV file or spreadsheets application file (e.g., *.xls, *.odt). Details for the data import have been described elsewhere (R Core Team, 2015; Rödiger et al., 2012). To keep the case study sections compact we have chosen to load datasets from the **qpcR** package (Ritz and Spiess, 2008) (v. 1.4.0) and the **RDML** package (v. 0.8-3) to our workspace. The **chipPCR** package (Rödiger et al., in press) (v. 0.0.8-10) was used for data pre-processing, quality control and the calculation of the quantification cycle (Cq).

The Cq is a quantitative measure, which represents the number of cycles needed to reach a user defined threshold signal level, in the exponential phase of a qPCR/qIA reaction. Several Cq methods have been described (Ruijter et al., 2013). In this study we have chosen the second derivative maximum method (Cq_{SDM}) and the 'Cycle threshold' (Cq_{Ct}) method.

During a perfect qPCR reaction, the target DNA doubles (2^n; n = cycle number) at each cycle. Here the amplification efficiency (AE) is 100 %. However, in reality, numerous factors cause an inhibition of the amplification ($AE < 100$ %). The AE can be determined by the relation of the Cq value depending on the sample input quantity as described in (Rödiger et al., in press; Svec et al., 2015).

In Rödiger et al. (2013a) we described the application of R for the analysis of melting curve experiments from microbead-based assays. We used functions from the **MBmca** package (0.0.3-4) for an analysis of the target specific melting temperature (T_M) in experiments of the present study, since the mathematical foundation is the same.

We completed our examples with case studies for the analysis of dPCR experiments. In particular, we used the **dpcR** package (v. 0.1.4.0) to estimate the number of molecules in a sample.

Results

In the following sections we will show how R can be used (I) as a unified open software for data analysis and presentation in research, (II) as software frame-work for novel technical developments, (III) as platform for teaching of new technologies and (IV) as reference for statistical methods.

Case study one – qPCR and amplification efficiency calculation

The goal of our first case study was to calculate the Cq values and the AE from a qPCR experiment. We used the `guescini1` dataset from the **qpcR** package, where the gene *NADH dehydrogenase 1* (MT-ND1) was amplified in a LightCycler® 480 (Roche) thermocycler. Details of the experiment are described in Guescini et al. (2008). We started with loading the required packages and datasets. A good practice for reproducible research is to track the package versions and environment used during the analysis. The function `sessionInfo` from the **utils** package provides this functionality. Assuming that the analysis starts with a clean R session it is possible to assign the required packages to an object, as shown below. Reproducibility of research can be further improved by the **archivist** package (Biecek and Kosinski, 2015), which stores and recovers crucial data and preserves metadata of saved objects (not shown). All settings of an R session can be easily saved and/or restored using the **settings** package (van der Loo, 2015).

```
# Load the required packages for the data import and analysis.
# Load the chipPCR package for the pre-processing and curve data quality
# analysis and load the qpcR package as data resource.
require(chipPCR)
require(qpcR)

# Collect information about the R session used for the analysis of the
# experiment.
current.session <- sessionInfo()

# Next, we load the 'guescini1' dataset from the qpcR package to the
# workspace and assign it to the object 'gue'.
gue <- guescini1

# Define the dilution of the sample DNA quantity for the calibration curve
# and assign it to the object 'dil'.
dil <- 10^(2: -4)
```

We previewed the amplification curve raw data using the `matplot` function (see code below). The amplification curve data showed a strong signal level variation in the plateau region (Figure 3A). Therefore, all data were subjected to a minimum-maximum normalization using the `CPP` function from the **chipPCR** package. In addition, all data were baselined and smoothed (Figure 3B; see Rödiger et al. 2013a; Rödiger et al. in press). The Cq values were calculated by the Cq_{SDM} and Cq_{Ct} methods as shown next.

```
# Pre-process the amplification curve data with the CPP function from the
# chipPCR package. The trans parameter was set TRUE to perform a baselining and
# the method.norm parameter was set to minm for a min-maximum normalization. All
# amplification curves were smoothed by Savitzky-Golay smoothing.

res.CPP <- cbind(gue[, 1], apply(gue[, -1], 2, function(x) {
  CPP(gue[, 1], x, trans = TRUE, method.norm = "minm", method.reg = "least",
      bg.range = c(1,7))[["y.norm"]]
}))

# Use the th.cyc function from the chipPCR package to calculate the Cq values
# by the cycle threshold method at a threshold signal level "r" of 0.05.
Cq.Ct <- apply(gue[, -1], 2, function(x)
  th.cyc(res.CPP[, 1], x, r = 0.05)[1])

# Use the inder function from the chipPCR package to calculate the Cq values
# by the SDM method. This will give a lot of output in the console.
Cq.SDM <- apply(gue[, -1], 2, function(x)
  summary(inder(res.CPP[, 1], x))[2])

# Fit a linear model to carry out a regression analysis.
res.Cq <- lm(Cq.Ct ~ Cq.SDM)
```

To compare the Cq_{SDM} and Cq_{Ct} methods we performed a regression analysis. The Cq methods are in a good agreement. However, the dispersion of the Cq_{Ct} values appeared to be higher than in the Cq_{SDM} values (Figure 3C).

```
summary(res.Cq)

Call:
lm(formula = Cq.Ct ~ Cq.SDM)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4904 -0.2730  0.0601  0.3540  1.1871

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -8.125534   0.207419  -39.17   <2e-16 ***
Cq.SDM       0.988504   0.008097  122.08   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5281 on 82 degrees of freedom
Multiple R-squared:  0.9945,  Adjusted R-squared:  0.9945
F-statistic: 1.49e+04 on 1 and 82 DF,  p-value: < 2.2e-16
```

The dilution ('dil') and the Cq ('Cq.Ct') values served as input for the calculation of the amplification efficiency (AE) with `effcalc` function from the **chipPCR** package. In our case study we needed to rearrange the 'Cq.Ct' values in a matrix using the command `effcalc(dil,t(matrix(Cq.Ct,nrow = 12,ncol = 7))`. For visualization of the confidence intervals of the regression analysis we set the parameter `CI = TRUE`.

```
# Arrange and plot the results in a convenient way.
layout(matrix(c(1, 2, 3, 3, 4, 5), 3, 2, byrow = TRUE))

# Store used margin parameters
def.mar <- par("mar")
layout(matrix(c(1, 2, 3, 3, 4, 5), 3, 2, byrow = TRUE))
# Set bigger top margin.
par(mar = c(5.1, 4.1, 6.1, 2.1))
```

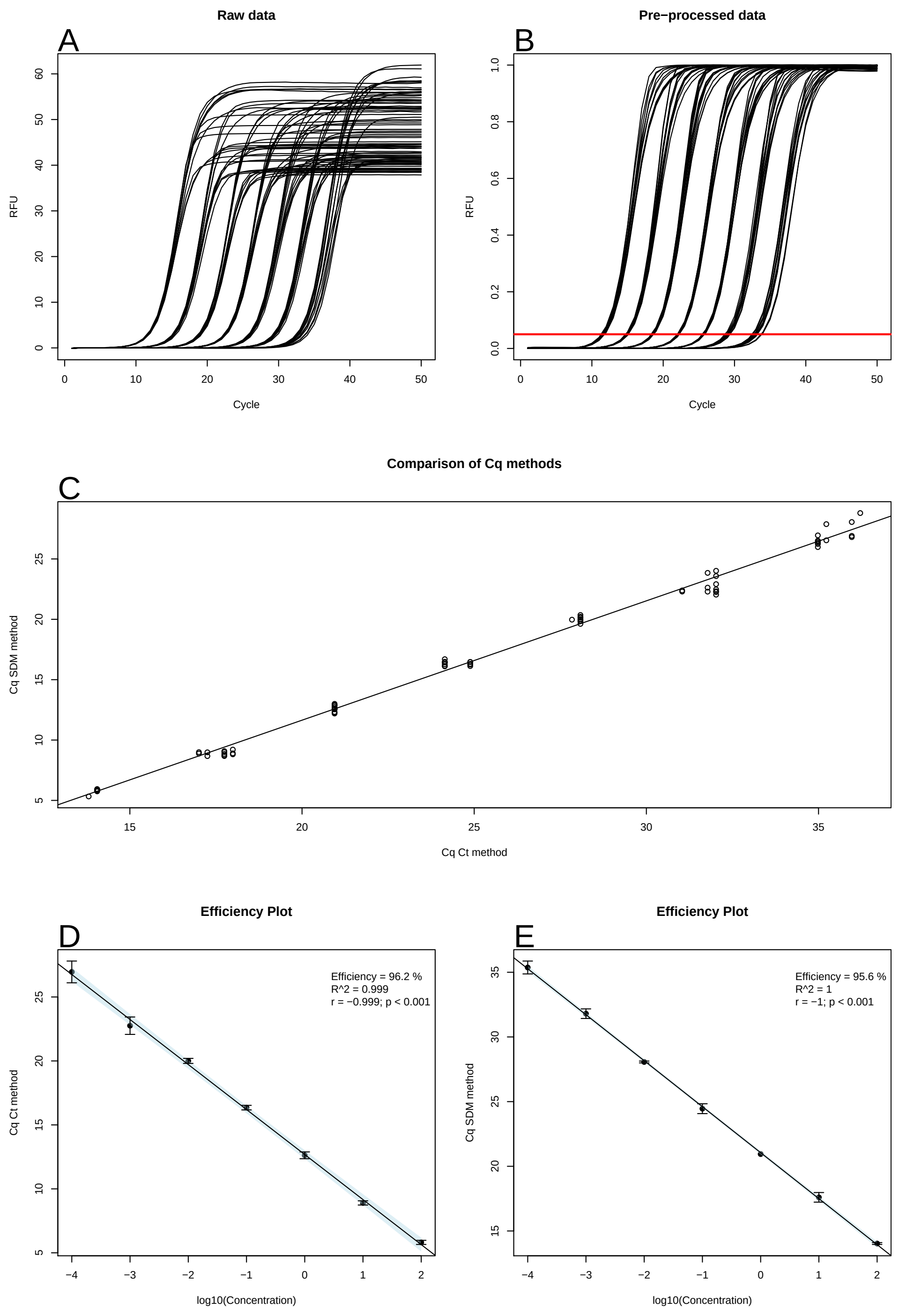


Figure 3: Analysis of the amplification curve data of the `guescini1` dataset. **(A)** Raw data from the calibration curve samples were visually inspected. The qPCR curves display a broad variation in plateau fluorescence (38–62 RFU). **(B)** The `CPP` function from the **chipPCR** package was used to baseline the data, to smooth the data with Savitzky-Golay smoothing filter and to normalize the data between 0 and 1. The red horizontal line (—) indicates the fluorescence level (0.05) used for the calculation of the Cq value by the 'cycle threshold' method. **(C)** The Cq values were calculated by the Cq_{SDM} method ('SDM method') (`inder`, **chipPCR**) and the Cq_{Ct} method ('Ct method') (`th.cyc`, **chipPCR**). The threshold value was set to $r = 0.05$. The Cq_{SDM} and Cq_{Ct} values were plotted and analysed by a linear regression ($R^2 = 0.9945$; $P < 2.2^{-16}$) and Pearson's r ($r = 0.9972605$; $P < 2.2^{-16}$). The amplification efficiency based on **(D)** Cq_{Ct} values and **(E)** Cq_{SDM} values were automatically analysed with the `effcalc` (**chipPCR**) function. Cq, Quantification cycle; SDM, Second derivative maximum, R^2, Coefficient of determination; r, Pearson product-moment correlation coefficient, RFU, relative fluorescence units.

```
# Plot the raw amplification curve data.
matplot(gue[, -1], type = "l", lty = 1, col = 1, xlab = "Cycle",
        ylab = "RFU", main = "Raw data")
mtext("A", side = 3, adj = 0, cex = 2)

# Plot the pre-processed amplification curve data.
matplot(res.CPP[, -1], type = "l", lty = 1, col = 1, xlab = "Cycle",
        ylab = "RFU (normalized)", main = "Pre-processed data")
mtext("B", side = 3, adj = 0, cex = 2)
abline(h = 0.05, col = "red", lwd = 2)

# Plot Cq.SDM against Cq.Ct and add the trendline from the linear regression
# analysis.

plot(Cq.SDM, Cq.Ct, xlab = "Cq Ct method", ylab = "Cq SDM method",
     main = "Comparison of Cq methods")
abline(res.Cq)
mtext("C", side = 3, adj = 0, cex = 2)

# Use the effcalc function from the chipPCR package to calculate the
# amplification efficiency.
plot(effcalc(dil, t(matrix(Cq.Ct, nrow = 12, ncol = 7))), ylab = "Cq Ct method",
     CI = TRUE)
mtext("D", side = 3, adj = 0, cex = 2)

plot(effcalc(dil, t(matrix(Cq.SDM, nrow = 12, ncol = 7))), ylab = "Cq SDM method",
     CI = TRUE)
mtext("E", side = 3, adj = 0, cex = 2)

# Resore margin default values.
par(mar = c(5.1, 4.1, 4.1, 2.1))
```

Finally, Cq values were plotted using the `layout` function (Figures 3D and E). The Cq values and amplification vary slightly between both methods. This is an expected observation and is in accordance to the findings by Ruijter et al. (2013). As shown in this case study, it is easy to set-up a streamlined workflow for data read-in, pre-processing and analysis with a few functions.

Case study two – qPCR and melting curve analysis

A common task during the analysis of qPCR experiments is to distinguish between positive and negative samples (see Figure 5). If the melting temperature of a sample is known it is possible to automatize the decision by a melting curve analysis (MCA). As shown in Rödiger et al. (2013a) this can be done by interrogating the T_M. Therefore, we used a logical statement, which tests if T_M is within a tight temperature range. We used the signal height as second parameter. In line with 'Case study one – qPCR and amplification efficiency calculation we used the function `sessionInfo` to track all packages used during the analysis. Reproducible research is greatly enhanced if open data exchange formats are used. Therefore, we used the **RDML** package for data read-in. The amplification and melting curve data were measured with a CFX96 system (Bio-Rad) and then exported as RDML v 1.1 format file as 'BioRad_qPCR_melt.rdml'. Within this qPCR experiment we amplified the *Mycobacterium tuberculosis katG* gene and tried to detect a mutation at codon 315. The experiment was separated in two parts:

1. Detection of overall *M. tuberculosis* DNA (wild-type and mutant) and
2. specific detection of wild-type *M. tuberculosis* by melting of TaqMan probe (quencher – BHQ2, fluorescent reporter – Cy5) with amplified DNA (see Luo et al. 2011 for probe/primer sequences and further details).

The qPCR was conducted using EvaGreen® Master Mix (Syntol) according to the manufacturer's instructions, with 500 nM of primers and probe in a 25 μL final reaction volume. Thermocycling was conducted using a CFX96 (BioRad) initiated by 3 min incubation at 95 °C, followed by 41 cycles (15 s at 95 °C; 40 s at 65 °C) with a single read-out taken at the end of each cycle. Probe melting was conducted between 35 °C and 95 °C by 1 °C at 1 s steps.

RDML file structures can be complex. Though RDML files are XML structured files and thus intended to be readable by humans, it is hard to grasp the complex hierarchical file structure without some basic understanding. A simple and fast method to compactly display the structure of an object in R is to use the `str` or `summary` function (not shown). However, such R tools are not informative in this context. Therefore we implemented a dendrogram-like view (Figure 4). According to this, the file contains different datasets, each with 3 samples, ('pos', 'ntc', 'unknown'). Only a subset of the data was used in our case study and combined to the object `qPCR`.

```
# Import the qPCR and melting curve data via the RDML package.
# Load the chipPCR package for the pre-processing and curve data quality
# analysis and the MBmca package for the melting curve analysis.
require(RDML)
require(chipPCR)
require(MBmca)
require(dplyr)

# Collect information about the R session used for the analysis of the qPCR
# experiment.
current.session <- sessionInfo()

# Load the BioRad_qPCR_melt.rdml file from the RDML package and assign the data
# to the object 'BioRad'.
filename <- file.path(path.package("RDML"), "extdata", "BioRad_qPCR_melt.rdml")
BioRad <- RDML$new(filename)

# The structure of the file can be overviewed by the AsDendrogram() function.
# We can see that our experiment contains two detection channels (Figure 4).
# ('EvaGreen' and 'Cy5' at 'Run ID'). 'EvaGreen' channel has one
# probe (target) - 'EvaGreen'. 'Cy5' has: 'Cy5', 'Cy5-2' and 'Cy5-2_rr'.
# each target has three sample types (positive, unknown, negative).
# And each sample type has qPCR ('adp') and melting ('mdp') data.
# The last column shows the number of samples in an experiment.

BioRad$AsDendrogram()

# Fetch cycle dependent fluorescence for the EvaGreen channel and row 'D'
# (contains the target 'Cy5-2' in the channel 'Cy5') of the
# katG gene and aggregate the data in the object 'qPCR'.

qPCR <- BioRad$AsTable() %>%
  filter(target == "EvaGreen",
         grepl("^D", position))  %>% BioRad$GetFData(.)
```

We inspected and pre-processed a subset of the amplification curve data solely using functionalities provided by the **chipPCR** package. The `plotCurves` function was used to get an overview of the curvatures. The green background colour of subplots shows that the data contain no missing values. Some curves indicated a moderate baseline shift with a slight negative trend (Figure 5). This observation suggested to baseline the raw data by using a linear regression model (cycles x - y; ($bg.range = c(x, y)$) in the `CPP` function.). The curvatures of 'D1_Alm12...' and 'D2_Alm12...' exhibited a drop in the plateau phase. However, this is not of importance during this stage of the analysis.

```
# Use plotCurves function to get an overview of the amplification curve samples
# (Figure 5).

plotCurves(qPCR[, 1], qPCR[, -1], type = "l")
mtext("Cycles", SOUTH <- 1, line = 3)
mtext("Fluorescence", side = 2, line = 2)
```

Since the amplification curves indicated that selected samples (except non-template-control; 'NTC') are positive, we distinguished between true positive and true negative samples by MCA (Figure 6A).

```
# To detect positive samples we calculated the Cq values by the cycle threshold
# method. This method is implemented in the th.cyc function. The threshold signal
# level r was set to 10.
Cq.Positive <- t(apply(qPCR[, -1], 2, function(x)
{
  res <- CPP(qPCR[, 1], x, trans = TRUE, bg.range = c(2, 8),
             method.reg = "least")[["y.norm"]]
```

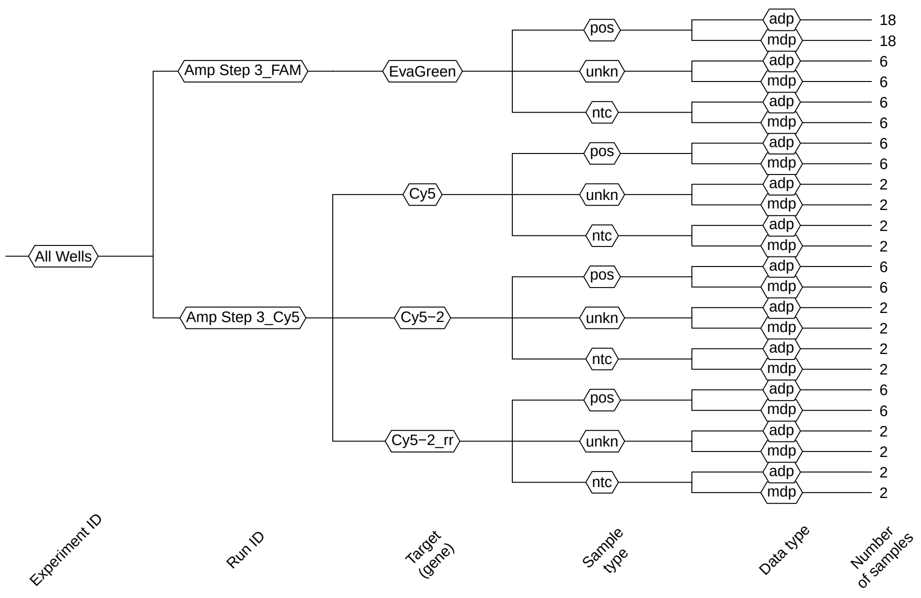


Figure 4: File structure visualization of the RDML file 'BioRad_qPCR_melt.rdml' form the **RDML** package. The file was read by the `RDML` function and the structure displayed as dendrogram by the call `BioRad$AsDendrogram()`. Names are used according to the RDML convention by Lefever et al. (2009). The object `BioRad` branches into an experiment with `Run ID` names for two fluorescence detection channels (FAM, Cy5). The targets have typical designations like `pos` (positive), `unkn` (unknown) and `ntc` (non-template control). In the deeper branches are the data types `adp` (amplification data point) and `mdp` (melting data point) shown with the number of samples (ranging from 2 to 18).

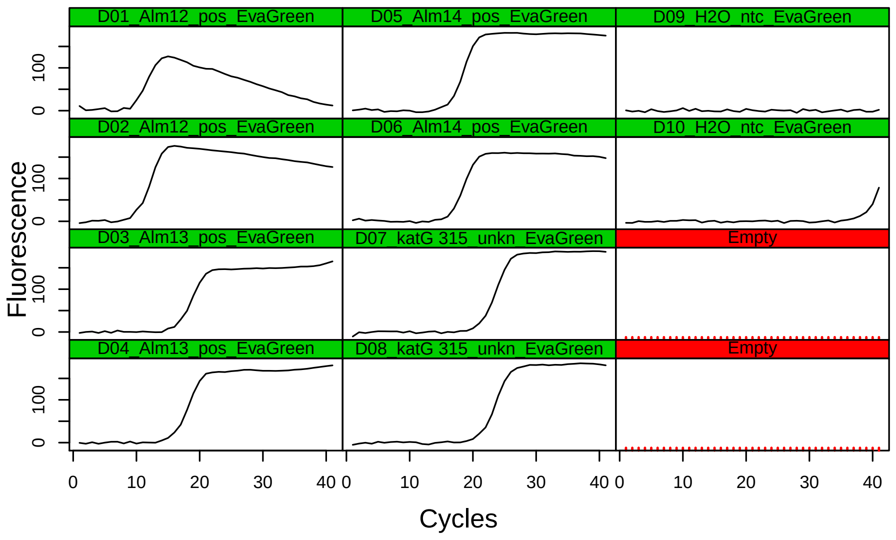


Figure 5: Analysis of the amplification curve data. The calibration curve samples were inspected by the `plotCurves` function from the **chipPCR** package. The green colour code indicates that the data contain no missing values. However, the visual inspection revealed that the data are noisy. All samples ('D1_Alm12...'–'D8_Alm12...') appeared to be positive. One negative control ('D10_H20_ntc_EvaGreen') seems to be contaminated.

```
# The th.cyc fails when the threshold exceeds a maximum observed fluorescence
# value. Therefore, we used try() to allow an error-recovery.

th.cycle <- try(th.cyc(qPCR[, 1], res, r = 10, linear = FALSE)[1], silent = TRUE)

# In addition we used logical statements, which define if a
cq <- ifelse(is(th.cycle, "try-error"), as.numeric(th.cycle), NA)
if(th.cycle > 35) cq <- NA
```

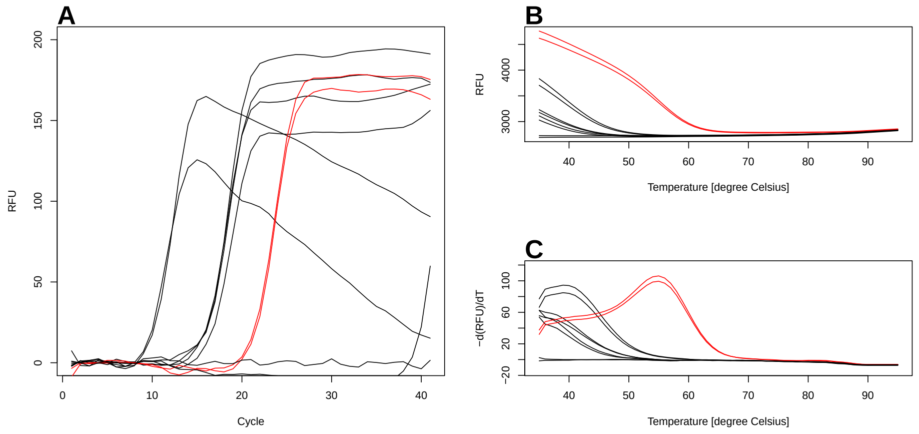


Figure 6: Amplification curve plots and melting curve plots. **(A)** The raw amplification curve data were pre-processed with the CPP function prior to visualization. To calculate the T_M values from the raw melting curve data **(B)** we used the diffQ function from the **MBmca** package. **(C)** We adjusted our algorithm to plot the true positive melting peaks in red, while negative melting peaks were labelled in black. The inspection of the plot and the output of results.tab showed that only D07_katG 315_unkn_EvaGreen (—) and D08_katG 315_unkn_EvaGreen (—) are true positive. RFU, relative fluorescence units; -d(RFU)/dT, negative first derivative of the melting-curve.

```
  pos <- !is.na(cq)
  c(Cq = cq, M.Tub_positive = pos)
}
))
```

So far we were able to calculate the Cq values for all samples. Next, we performed a melting curve analysis to characterize our samples. Therefore, we fetched the melting curve data from the BioRad object. We focused on the central steps of the melting curve analysis. A comprehensive overview on such analysis was presented in Rödiger et al. (2013a).

```
# Fetch temperature dependent fluorescence for the Cy5 channel of the
# probe 'Cy5-2' that can hybridize with Mycobacterium tuberculosis
# katG gene (codon 315) and aggregate the data in the object 'melt'.
melt <- BioRad$AsTable() %>%
  filter(target == "Cy5-2")  %>% BioRad$GetFData(., data.type = "mdp")

# Calculate the melting temperature with the diffQ function from the MBmca
# package. Use simple logical conditions to find out if a positive sample with
# the expected Tm of circa 54.5 degree Celsius is found. The result of the test
# is assigned to the object 'Tm.Positive'.

Tm.Positive <- matrix(nrow = ncol(melt) - 1,
                      byrow = TRUE,
                      dimnames = list(colnames(melt)[-1]),
                      unlist(apply(melt[, -1], 2, function(x) {
                        res.Tm <- diffQ(cbind(melt[, 1], x),
                                        fct = max, inder = TRUE)
                        positive <- ifelse(res.Tm[1] > 54 &
                                           res.Tm[1] < 55 &
                                           res.Tm[2] > 80, 1, 0)
                        c(res.Tm[1], res.Tm[2], positive)
                      })))
```

An important part of an automatized report generation is the automatic assignment of decisions (e.g., sample is a wild-type or mutant). Therefore, we used again a simple logic.

```
# Present the results in a tabular output as data.frame 'results.tab'.
# Result of analysis logic is:
# Cq.Positive && Tm.Positive = Wild-type
# Cq.Positive && !Tm.Positive = Mutant
# !Cq.Positive && !Tm.Positive = NTC
# !Cq.Positive && Tm.Positive = Error

results <- sapply(1:length(Cq.Positive[,1]), function(i) {
  if(Cq.Positive[i, 2] == 1 && Tm.Positive[i, 3] == 1)
    return("Wild-type")
  if(Cq.Positive[i, 2] == 1 && Tm.Positive[i, 3] == 0)
    return("Mutant")
  if(Cq.Positive[i, 2] == 0 && Tm.Positive[i, 3] == 0)
    return("NTC")
  if(Cq.Positive[i, 2] == 0 && Tm.Positive[i, 3] == 1)
    return("Error")
})
```

We applied names of variables and factors to all the objects (e.g, `Tm.Positive`, `Cq.Positive`) and aggregated them in the object `results.tab`. The aim of doing this is to present the result in an easy readable form.

```
results.tab <- data.frame(cbind(Cq.Positive, Tm.Positive, results))
names(results.tab) <- c("Cq", "M.Tub DNA", "Tm", "Height",
                        "Tm positive", "Result")

results.tab[["M.Tub DNA"]] <- factor(results.tab[["M.Tub DNA"]],
                                     labels = c("Not Detected", "Detected"))

results.tab[["Tm positive"]] <- factor(results.tab[["Tm positive"]],
                                       labels = c(TRUE, FALSE))
results.tab
```

The results of the analysis can be invoked by the statement `results.tab` (not shown). Finally, we plotted and printed the output of our melting curve (Figure 6B) and melting peak (Figure 6C) analysis.

```
# Convert the decision from the 'results' object in a colour code:
# Negative, black; Positive, red.

color <- c(Tm.Positive[, 3] + 1)

# Arrange the results of the calculations in a plot.
layout(matrix(c(1, 2, 1, 3), 2, 2, byrow = TRUE))

# Use the CPP function to preporcess the amplification curve data.
plot(NA, NA, xlim = c(1, 41), ylim = c(0,200), xlab = "Cycle", ylab = "RFU")
mtext("A", cex = 2, side = 3, adj = 0, font = 2)
lapply(2L:ncol(qPCR), function(i)
  lines(qPCR[, 1],
        CPP(qPCR[, 1], qPCR[, i], trans = TRUE,
            bg.range = c(1,9))[["y.norm"]],
        col = color[i - 1]))

matplot(melt[, 1], melt[, -1], type = "l", col = color,
        lty = 1, xlab = "Temperature [degree Celsius]", ylab = "RFU")
mtext("B", cex = 2, side = 3, adj = 0, font = 2)

plot(NA, NA, xlim = c(35, 95), ylim = c(-15, 120),
     xlab = "Temperature [degree Celsius]",
     ylab = "-d(RFU)/dT")
mtext("C", cex = 2, side = 3, adj = 0, font = 2)

invisible(
  lapply(2L:ncol(melt), function(i)
    lines(diffQ(cbind(melt[, 1], melt[, i]), verbose = TRUE,
                fct = max, inder = TRUE)[["xy"]], col = color[i - 1]))
)
```

According to the analysis by MCA the samples 'D07_katG315. . . ' and 'D08_katG315. . . ' were the only positive samples. The remaining samples appeared to be positive in the amplification plot in Figure 5 due to primer-dimer formation or sample contamination.

Case study three – Isothermal amplification

Isothermal amplification is an alternative nucleic acid amplification method, which uses a constant temperature rather than cycling through denaturation, annealing and extension steps (Rödiger et al., 2014). The signal is monitored continuously on a time-wise basis. In qIA the abscissa values are often not uniformly spaced. Our CPP function gives a warning in such a case. In qIA the Cq values are dependent on the time instead of cycles. In this study we used the th.cyc function from the **chipPCR** package to determine the time (Cq_t) required to reach a defined threshold signal level.

We performed a quantitative isothermal amplification (qIA) with the plasmid *pCNG1* by using a Helicase dependent amplification (HDA). Our previously reported VideoScan platform (Rödiger et al., 2013b) was used to control the temperature and to monitor the amplification reaction. The VideoScan technology is based on a highly versatile fluorescence microscope imaging platform, which can be operated with a heating/cooling unit (HCU) for qPCR and MCA applications (Rödiger et al., 2013a,b; Spiess et al., 2015). Since the enzyme DNA Helicase unwinds DNA, no thermal denaturation is needed. HDA conditions were taken from the 'IsoAmp III Universal tHDA Kit', Biohelix Corp, as described by the vendor. In detail, the reaction was composed of 'mix A' 10 μL A. bidest, 1.25 μL 10X buffer, 0.75 μL primer (150 nM final), 0.5 μL template plasmid. Preincubation: The mixture was incubated for 2 min at 95°C and immediately placed on ice. Reaction 'mix B' contained 5 μL A. bidest., 1.25 μL 10X buffer, 2 μL NaCl, 1.25 μL $MgSO_4$, 1.75 μL dNTPs, 0.25 μL EvaGreen (Biotium), 1 μL enzyme mix. The mix was covered with 50 μL mineral oil (Roth). The fluorescence measurement in VideoScan HCU started directly after adding 'mix B' at 65°C. A $1x$ (D1) and a $1:10$ dilution (D2) were tested. The resulting dataset C81 is part of the **chipPCR** package. Two concentrations (stock and 1:10 diluted stock) of input DNA were used in the HDA. First, we had a look at the C81 dataset with the str function.

```
# This case study uses the qIA raw data (C81 dataset) from the chipPCR
# package. Therefore, first we load the chipPCR package.
require(chipPCR)

# To see the structure of the C81 dataset we used the str function.
str(C81)
'data.frame':        351 obs. of  5 variables:
 $ Cycle : int  0 1 2 3 4 5 6 7 8 9 ...
 $ t.D1  : int  0 51 73 90 107 124 140 157 174 190 ...
 $ MFI.D1: num  0.549 0.535 0.532 0.53 0.525 ...
 $ t.D2  : int  0 19 53 72 91 110 128 147 166 185 ...
 $ MFI.D2: num  0.77 0.523 0.514 0.51 0.508 ...
```

C81 is a data frame. The first column contains the measuring points (Cycle) and the consecutive columns contain the time stamps (e.g., t.D1, in seconds according to the **chipPCR** manual) and the signal hight (MFI.D1, as mean fluorescence intensity; MFI). A brief look at the time stamps showed that the data are not uniformly spaced. Warning messages by the CPP functions are just a reminder for the user to take care during the pre-processing. Methods like smoothing might cause artifacts (Spiess et al., 2015). We know that the HDA is a time-dependent reaction. Therefore, we do not have a use for the discrete measuring points. Next, we proceeded with the analysis. Similar to the previous case studies, we prepared the plot of the raw data. Since the raw data showed a slight negative trend and an off-set of circa 0.45 MFI (mean fluorescence intensity) it was necessary to pre-process the raw data (Figure 7A).

```
# Drawn in an 2-by-1 array on the device by two columns and one row.
par(mfrow = c(1, 2))

# Plot the raw data from the C81 dataset to the first array and add
# a legend. Note: The abcsissa values (time in seconds) were divided
# by 60 (C81[, i] / 60) to convert to minutes.
plot(NA, NA, xlim = c(0, 120), ylim = c(0, 1.2), xlab = "Time (min)", ylab = "MFI")
mtext("A", cex = 2, side = 3, adj = 0, font = 2)
lapply(c(2, 4), function(i) {
  lines(C81[, i] / 60, C81[, i + 1], type = "b", pch = 20, col = i - 1)
})
legend("topleft", c("D1: 1x", "D2: 1:10 diluted sample"), pch = 19, col = c(1, 3),
       bty = "n")

# Prepare a plot on the second array for the pre-processed data.
plot(NA, NA, xlim = c(0, 120), ylim = c(0, 1.2), xlab = "Time (min)", ylab = "MFI")
mtext("B", cex = 2, side = 3, adj = 0, font = 2)
```

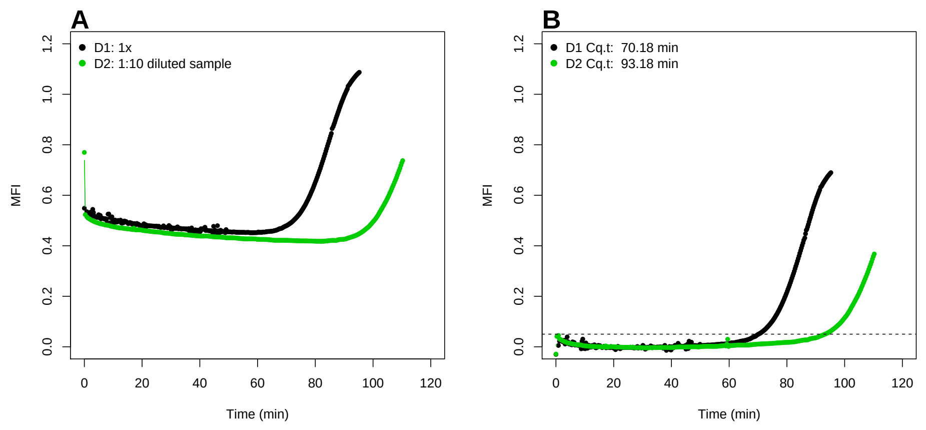


Figure 7: Quantitative isothermal amplification by Helicase dependent amplification (HDA). **(A)** The raw data of the HDA (D1, undiluted; D2, 1 : 10 diluted) exhibit some outliers (detector artifacts), an off-set of circa 0.5 MFI and a slight negative trend in the baseline region (0–52 minutes). **(B)** We used the `CPP` function to smooth the data with a spline function. Baselining was done with a linear regression model (robust MM-estimator). Finally, we used the `th.cyc` function (**chipPCR**) to calculate the cycle threshold time for samples D1 and D2. The threshold value was set to $r = 0.05$ ($--$, threshold line). Cq_t, required time to reach a defined threshold signal level. MFI, mean fluorescence intensity.

First, we used the `CPP` function to pre-process the raw data. Similar to the other case studies we baselined and smoothed the amplification curve data prior to the analysis of the of the Cq_t value. However, instead of the Savitzky-Golay smoother we used a cubic spline (`method = "spline"`) in the `CPP` function. In addition, outliers were automatically removed in the baseline region (Figure 7A and B). The background range was defined by bare eye to be between the 1^{st} and 190^{th} data point (corresponding to a baseline region between 0 and 52 minutes). The `CPP` uses a linear least squares or robust fit methods (e.g., Rfit by Kloke and McKean 2012) to estimate the slope of the background (Rödiger et al., in press).

```
# Apply the CPP functions to pro-process the raw data.1) Baseline data to zero,
# 2) Smooth data with a spline, 3) Remove outliers in background range between
# entry 1 and 190. Assign the results of the analysis to the object 'res'.
res <- lapply(c(2, 4), function(i) {
  y.s <- CPP(C81[, i] / 60, C81[, i + 1],
             trans = TRUE,
             method = "spline",
             bg.outliers = TRUE,
             bg.range = c(1, 190))
  lines(C81[, i] / 60, y.s[["y.norm"]], type = "b", pch = 20, col = i - 1)
  # Use the th.cyc function to calculate the cycle threshold time (Cq.t).
  # The threshold signal level r was set to 0.05. NOTE: The function th.cyc
  # will give a warning in case data are not equidistant. This is intentional
  # to make the user aware of potential artificats due to pre-processing.
  paste(round(th.cyc(C81[, i] / 60, y.s[["y.norm"]], r = 0.05)[1], 2), "min")
})

# Add the cycle threshold time from the object 'res' to the plot.

abline(h = 0.05, lty = 2)
legend("topleft", paste(c("D1 Cq.t: ", "D2 Cq.t: "), res), pch = 19,
       col = c(1, 3), bty = "n")
```

The pre-processed data were subjected to the analysis of the Cq_t values. It is important to note that the trend correction and proper baseline was a requirement for a sound calculation. We calculated Cq_t values of 70.18 minutes and 93.18 minutes for the stock (D1) and 1:10 (D2) diluted samples, receptively.

Case study four – Digital PCR

We have developed the **dpcR** package for analysis and presentation of digital PCR experiments. This package can be used to build custom-made analysis pipelines and provides structures to be openly extended by the scientific community. Simulations and predictions of binomial and Poisson distributions, commonly used theoretical models of dPCR, statistical data analysis methods, plotting facilities and report generation tools are part of the package (Pabinger et al., 2014). Here, we show a case study for the **dpcR** package. Simulations are part in many educational curricula and greatly support teaching. In this case study, we mimicked an *in silico* experiment for a droplet digital PCR experiment similar to Figure 1. The aim was to assess the concentration of the template sample. In the following we will use the expression partition as synonym for droplet. The number of positive partitions (k), total number of partitions (n) and the size of the partition are the only data required for the analysis. The estimate of the mean number of template molecules per partition ($\hat{\lambda}$) was calculated using the following equation (Huggett et al., 2013):

$$\hat{\lambda} = -\ln(1 - \frac{k}{n}). \tag{1}$$

The average partition volume in our experiment was assumed to be 5 nL. We counted $n = 16800$ partitions in total from which $k = 4601$ partitions were positive. The binomial distribution of positive and negative partitions was used to determine $\hat{\lambda}$ (Figure 8). Our package allows both easy estimation of a density of the parameter and calculation of confidence intervals (CI) using Wilson's method (Brown et al., 2001) at a confidence interval level of 0.95. The true number of template molecules per partition ($\hat{\lambda}$) is likely to be included in the range of the CI (Milbury et al., 2014). The obtained mean number of template molecules per partition multiplied by the volume of the partitions ($16800 \cdot 5$ nL) constitutes the sample concentration.

```
# Load the dpcR package for the analysis of the digital PCR experiment.
require(dpcR)

# Analysis of a digital PCR experiment. The density estimation.
# In our in-silico experiment we counted in total 16800 partitions (n).
# Thereof, 4601 were positive (k).
k <- 4601
n <- 16800
(dens <- dpcr_density(k = k, n = n, average = TRUE, methods = "wilson",
                      conf.level = 0.95))
legend("topleft", paste("k:", k,"\nn:", n))
#dev.off()
# Let us assume, that every partition has roughly a volume of 5 nL.
# The total concentration (and its confidence interval) in molecules/ml is
# (the factor 1e-6 is used for the conversion from nL to mL):
dens[4:6] / 5 * 1e-6
```

Since we assumed a partition volume of 5 nL we have a total volume of 0.084 mL and 6.40×10^{-8} (95% CI: 6.2×10^{-8}–6.6×10^{-8}) molecules/mL in the sample.

Selected functionality was implemented as an interactive **shiny** (Chang et al., 2015) GUI application to make the software accessible for users who are not fluent in R and for experts who wish to automatize routine tasks. Details and examples of the **shiny** web application framework for R can be found at http://shiny.rstudio.com/. We implemented flexible user interfaces, which run the analyses and graphical representation into interactive web applications either as service on a web server or on a local machine without knowledge of HTML or ECMAScript (see the **dpcR** manual). The interface is designed in a cascade workflow approach (Data import → Analysis → Output → Export) with interactive user choices on input data, methods and parameters using typical GUI elements such as sliders, drop-downs and text fields. An example can be found at https://michbur.shinyapps.io/dpcr_density/. This approach enables the automatized output of R objects in combined plots, tables and summaries.

Case study five – Digital PCR partition volume correction

There is an ongoing debate in the scientific community about the effect of the partition volume on the estimated copy numbers size (Huggett et al., 2015a; Corbisier et al., 2015; Majumdar et al., 2015). Corbisier et al. (2015) showed that the partition volume is a critical parameter for the measurement of copy number concentrations. Their study revealed that the average droplet volume defined in the QuantaSoft software (v. 1.3.2.0, BioRad QX100 Droplet Digital PCR System) is 8 % lower than the real volume. In consequence, results of quantifications were systematically biased between different dPCR platforms. Case study four served as an introduction into the analysis of simulated dPCR experiments with the **dpcR** package. In the next case study, number five, we used the `pds_raw` dataset, which was generated by a BioRad QX100 Droplet Digital PCR System experiment. We re-analysed the data with

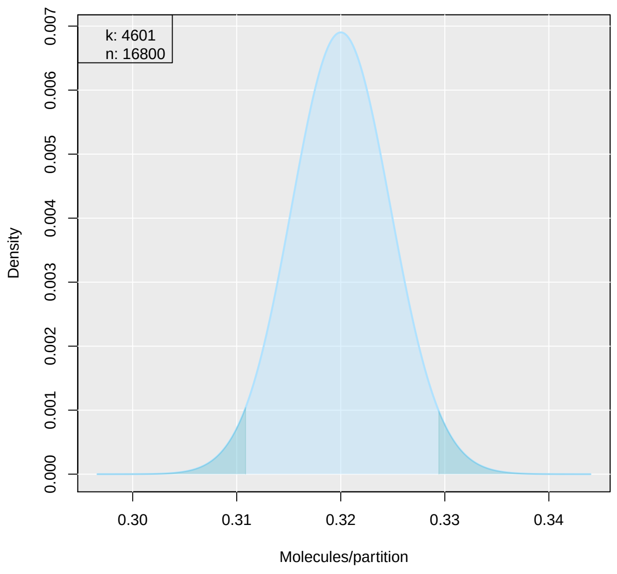


Figure 8: `dpcr_density` function from the **dpcR** package used for analysis of a droplet digital PCR experiment. From 16800 counted partitions (n) 4601 were positive (k). The chart presents the distribution of mean number of template molecules per partition ($\hat{\lambda}$). n, number of total partitions; k, number of positive partitions.

the partition volume of 0.834 nL as proposed by Corbisier et al. (2015) and a volume of 0.90072 nL as used in the BioRad QX100 Droplet Digital PCR System.

Our experimental setup was as follows. A duplex assay was used to simultaneously detect a constant amount of genomic DNA (theoretically 10^2 copies/μL) and a variable amount of plasmid DNA (10-fold serial dilution, not shown). The genomic DNA was isolated from *Pseudomonas putida KT2440* and the plasmid was *pCOM10-StyA::EGFP StyB*. Template DNA was heat treated at 95 °C for 5 min prior to PCR. We detected in Channel 1 the genomic DNA marker *ileS* with FAM labelled Taqman probes and in Channel 2 the plasmid DNA marker *styA* with HEX labelled Taqman probes. All primer/probe sequences and experimental condiditions are described in Jahn et al. 2013, 2014 and the manual of the **dpcR** package. Gating and partition clustering was taken without any modification from the data output of the BioRad QX100 Droplet Digital PCR System. Each partition is represented by a dot in Figure 9. First, we had a look at the data structure of the `pds_raw` dataset.

```
# Load the dpcR package and use the pds_raw dataset for the analysis of the
# digital PCR experiment.
# To get an overview of the data set we used the head and summary R functions
# in a chain. The output shows that the dataset contains lists of different
# samples (A01 ...)

require(dpcR)

head(summary(pds_raw))

    Length Class        Mode
A01 "3"    "data.frame" "list"
A02 "3"    "data.frame" "list"
A03 "3"    "data.frame" "list"
A04 "3"    "data.frame" "list"
B01 "3"    "data.frame" "list"
B02 "3"    "data.frame" "list"

# Next, we used str for the element A01. The element of the list contains a data frame
# with three columns. Two contain amplitude values (fluorescence intensity) and one
# contains cluster results (integer values of 1 - 4).

str(pds_raw[["A01"]])
'data.frame':        11964 obs. of  3 variables:
 $ Assay1.Amplitude: num  397 399 402 416 417 ...
 $ Assay2.Amplitude: num  3732 3808 4007 3778 3685 ...
 $ Cluster         : int  4 4 4 4 4 4 4 4 4 4 ...
```

Since the structure of the dataset was known now we selected samples for the analysis. According to the **dpcR** manual, the samples A02, B02, C02 and D02 contained the values for the replicates at a 1 : 100 dilution and G04 contained the values for the non-template control.

```
# Select the wells for the analysis. A02 to D02 are four replicate dPCR reactions
# and G04 is the no template control (NTC) (see dpcR manual for details).
wells <- c("A02", "B02", "C02", "D02", "G04")

# Set the arrangement for the plots. The first column contains the amplitude
# plots, column two the density functions and column three the concentration
# calculated according to the droplet volume as defined in the QX100 system,
# or the method proposed by Corbisier et al. (2015).
par(mfrow = c(5, 3))

# The function bioamp was used in a loop to extract the number of positive and
# negative partitions from the sample files. The results were assigned to the
# object 'res' and plotted.

for (i in 1L:length(wells)) {
  cluster.info <- unique(pds_raw[wells[i]][[1]]["Cluster"])
  res <- bioamp(data = pds_raw[wells[i]][[1]], amp_x = 2, amp_y = 1,
                main = paste("Well", wells[i]), xlab = "Amplitude of ileS (FAM)",
                ylab = "Amplitude of styA (HEX)", xlim = c(500,4700),
                ylim = c(0,3300), pch = 19)

  legend("topright", as.character(cluster.info[, 1]), col = cluster.info[, 1],
         pch = 19)
```

Next, we used the results from the object `res` to get the information about the number of positive partitions for the plasmid DNA marker *styA*. This is to be found in the clusters 2 and 3.

```
# Counts for the positive clusters 2 and 3 were assigned to new objects
# and further used by the function dpcr_density to calculate the number
# of molecules per partition and the confidence intervals. The results
# were plotted as density plot.

k.tmp <- sum(res[1, "Cluster.2"], res[1, "Cluster.3"])
# Counts for all clusters

n.tmp <- sum(res[1, ])

# Our next line is used to limit the x-axis of the plot to a meaningful range.
if(i < 5) x.lim <- c(0.065, 0.115) else x.lim <- c(0, 0.115)

# The next step is the calculation of the dPCR statistics.
dens <- dpcr_density(k = k.tmp, n = n.tmp, average = TRUE, methods = "wilson",
                     conf.level = 0.95, xlim = x.lim, bars = FALSE)
legend("topright", paste("k:", k.tmp,"\nn:", n.tmp), bty = "n")

# Finally, the concentration of the molecules was calculated with the volume
# used in the QX100 system and as proposed by Corbisier et al. (2015). The
# results were added as barplot with the confidence intervals.

res.conc <- rbind(original = dens[4:6] / 0.90072,
                  corrected = dens[4:6] / 0.834)
barplot(res.conc[, 1], col = c("white","grey"),
        names = c("Bio-Rad", "Corbisier"),
        main = "Influence of\nDroplet size", ylab = "molecules/nL",
        ylim = c(0, 1.5 * 10E-2))
arrows(c(0.7, 1.9), res.conc[, 2], c(0.7, 1.9), res.conc[, 3], angle = 90,
       code = 3, lwd = 2)
```

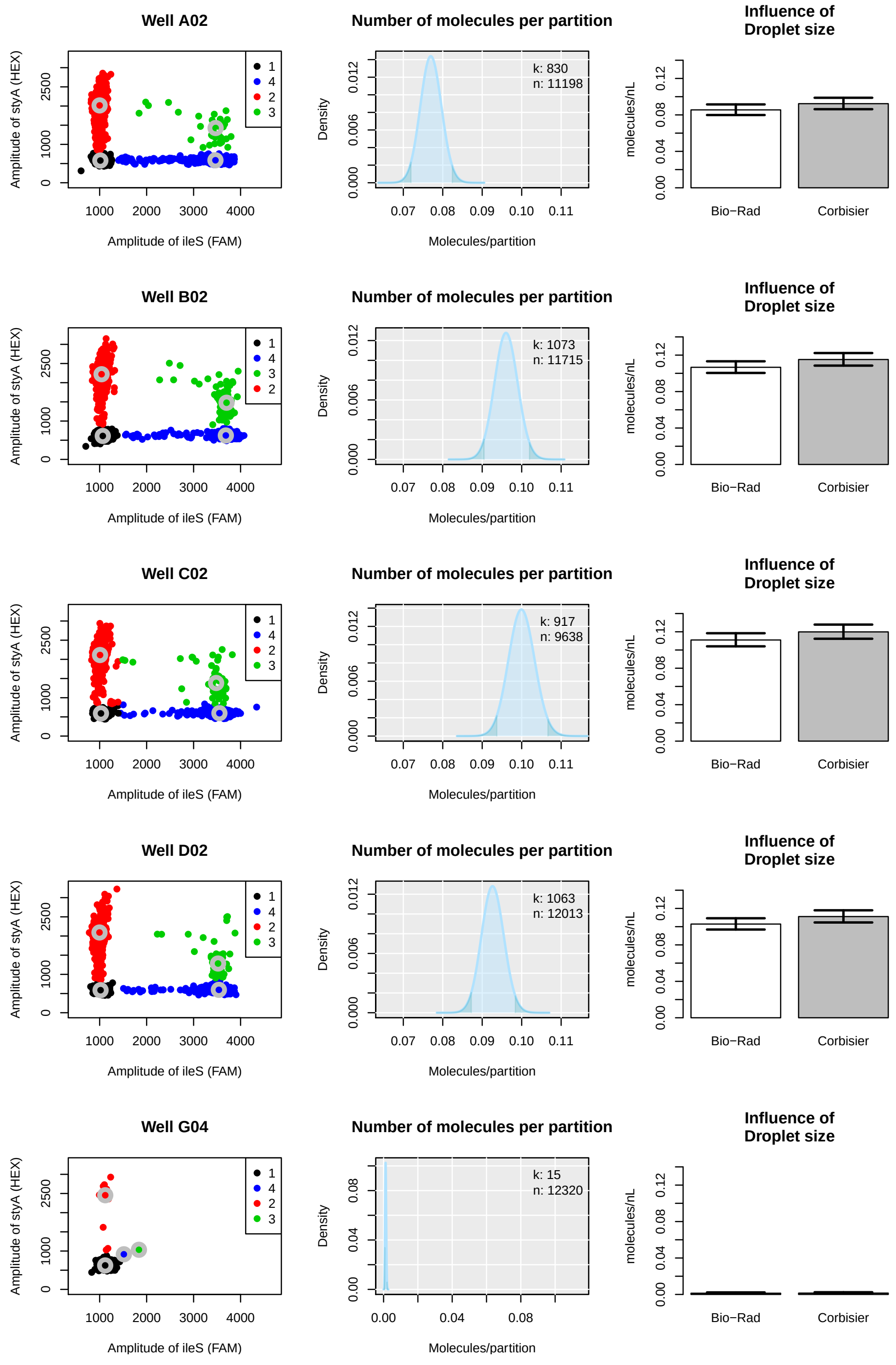


Figure 9: Analysis of a droplet digital PCR experiment. We used the `pds_raw` dataset from the **dpcR** package. All raw data were generated with a BioRad QX100 Droplet Digital PCR System. Column one: Amplitude plot of raw data shown by the `bioamp` function. Each dot represents a partition in a cluster. *Cluster* 1 •, *Cluster* 2 • = Target, *Cluster* 3 • = Target, *Cluster* 4 •. A02–D04 are replicate measurements and well G04 is a negative control; Column two: Density function with showing the number of molecules per partition. All replicates had similar numbers of counted positive partitions (k) and total number of partitions (n); Column three: Concentration of DNA molecules based on the volume used by the BioRad QX100 Droplet Digital PCR System and the volume proposed by Corbisier et al. (2015). n, number of total partitions; k, number of positive partitions; *styA*, styrene monooxygenase; *ileS*, isoleucyl-tRNA synthetase; FAM, Fluorescein channel; HEX, hexachlorofluorescein.

As proposed by Corbisier et al. (2015) the analysis with the non-corrected droplet volume results in an underestimation of sample concentrations (Figure 9 column 3). Our results show, that the replicates in Figure 9 A02 - D02 vary. However, the variation appeared to be within a similar range (Figure 9 column 2). The positive samples (Figure 9 A02–D02) clearly differ from the negative control

(Figure 9 G04). To compare this we performed a statistical test. This is basically a comparison of proportions as described elsewhere (Wang and Shan, 2013). Statistical capabilities of R simplify the next steps of the analysis of digital PCR experiments. The **dpcR** package offers a flexible wrapper `test_counts` around several methods of comparing results of several reactions. In the following, we used a test implemented in the **rateratio.test** package by Fay (2010).

```
# The first step is as usual extracting data and shaping it into an object of
# appropriate class, in this case 'ddpcr' (droplet digital PCR)

cluster_data <- do.call(bind_dpcr, lapply(1L:length(wells), function(i) {
  cluster.info <- unique(pds_raw[wells[i]][[1]]["Cluster"])
  res <- bioamp(data = pds_raw[wells[i]][[1]], amp_x = 2, amp_y = 1, plot = FALSE)
  # create ddpcr object for each experiment
  create_dpcr(data = c(rep(1, res[1, "Cluster.2"]), rep(1, res[1, "Cluster.3"])),
                n = as.integer(sum(res[1, ])), threshold = 1,
              type = "np", adpcr = FALSE)
}))

# Message: 'Different number of partitions.' is expected while joining objects
# with uneven length as droplet-based experiments. The message is specifically
# verbose to also deliver that the bind_dpcr function does not
# recycle shorter vectors to prevent the addition of non-existent data points.

# Give experiments proper names.
colnames(cluster_data) <- wells

# We choose the ratio model, which uses multiple ratio tests from the 'rateratio.test'
# package.

comp <- test_counts(cluster_data, model = "ratio")
```

Output format of `test_counts` is familiar to all R users:

```
> comp

Groups:
    group      lambda   lambda.low   lambda.up
A02     b 0.077011051 0.0704320674 0.084179123
B02     c 0.096061636 0.0888030003 0.103883369
C02     c 0.099979708 0.0918318731 0.108811947
D02     c 0.092649940 0.0856180946 0.100230919
G04     a 0.001218274 0.0006348818 0.002337119

Results of multiple comparison:
             X_squared       p_value signif
B02 - A02   22.7202237  3.123089e-06    ***
C02 - A02   29.5319827  1.100031e-07    ***
D02 - A02   15.7795663  1.016671e-04    ***
G04 - A02  897.9635582 6.799223e-197    ***
C02 - B02    0.7480551  4.164049e-01
D02 - B02    0.6604392  4.164049e-01
G04 - B02 1132.7471108 1.260524e-247    ***
D02 - C02    2.7724424  1.198747e-01
G04 - C02 1171.4948655 9.557011e-256    ***
G04 - D02 1092.0273410 5.950879e-239    ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Model used:
[1] "ratio"

> summary(comp)

  group      lambda   lambda.low   lambda.up
1     a 0.001218274 0.0006348818 0.002337119
```

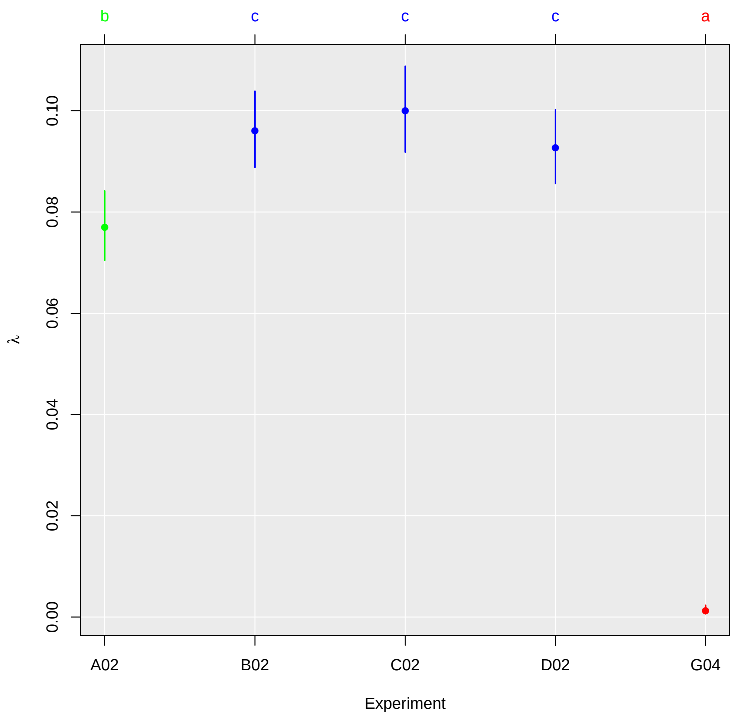


Figure 10: Comparison of droplet digital PCR runs. The `plot` method of the `test_counts` function from the **dpcR** package was used. The experiments were aggregated into colour coded groups based on their estimated mean number of copies per partition (λ). A02 remained individual b, B02, C02 and D02 formed a group c and the negative control also remained individual a.

```
2      b 0.077011051 0.0704320674 0.084179123
3      c 0.096230428 0.0887509893 0.104308745
```

The result of the analysis suggests that replicate samples B02, C02 and D02, with exception of A02, do not differ significantly. The negative control G04 was totally different from the other samples (Figure 10). Our **dpcR** package can be used to analyse data beyond a level as provided by commercial software. Access to the whole R environment allows performing more elaborated research in a highly customized workflow. In conclusion, our case study shows, that the R environment can be used to circumvent problems of locked-in systems.

Discussion and conclusion

This study gave a brief introduction to the analysis of qPCR, qIA, MCA and dPCR experiments with R. In addition to this, we briefly referenced to a vast collection of additional packages available from CRAN and Bioconductor. The packages may be considered as the building blocks (libraries) to create what users want and need. We showed that automatized research with R offers powerful means for statistical analysis and visualization. This software is not tied to a vendor or application (e.g., chamber or droplet based digital PCR, capillary or plate qPCR). It should be quite easy even for an inexperienced user to define a workflow and to set up a cross-platform environment for specific needs in a broad range of technical settings (Figure 11). This environment enforces no monolithic integration. We claim that the modular structure allows the user to perform flexible data analysis, adjusted to their needs and to design frameworks for high-throughput analysis. Furthermore, R enables the users to access and reuse code for the creation of reports in various formats (e.g., HTML, PDF).

Despite the fact that R is free of charge, it is quite possible to build commercial applications. The packages cover implementation of novel approaches and peer-reviewed analysis methods. R packages are an open environment to adopt to the growing knowledge in life sciences and medical sciences. Therefore, we argue that this environment may provide a structure for standardized nomenclature and serve as reference in qPCR and dPCR analysis. Speaking about openness, it needs to be emphasized that the main advantage of this software is its transparency at any time for anybody. Thus, it is possible to track numerical errors. A disadvantage of R is the lack of comprehensive GUIs for qPCR analysis. GUIs are key technologies to spread the use of R in bioanalytical sciences. Currently, we are establishing the ‘pcRuniveRsum’ (`http://michbur.github.io/pcRuniveRsum/`) as an on-line resource for the interested users. The command-line structure makes R ’inaccessible’ for many novices. We try

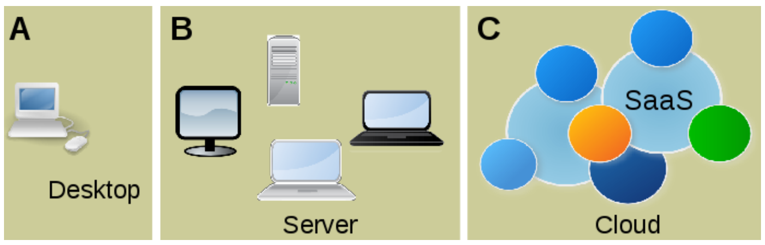


Figure 11: Deployment of R applications for the qPCR and dPCR experiments. **(A)** R is typically run from a desktop computer and operated by a GUI/IDE application such as RStudio or RKWard. This approach provides a flexible workflow for individuals. **(B)** Another approach is to run R with specific applications on a local server. Such scenarios are useful for the deployment within research departments or cooperate units (Nolan and Temple, 2014). **(C)** Cloud computing (CC) provides shared and scalable computing capacity (e.g., computing capacity, application software) and storage capacity (e.g., databases) as a service to an individual user or a community. Service categories include: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) over a network. Providers of CC manage the infrastructure and resources to achieve coherence and economies of scale similar to a utility over a network (i.p., Internet) (Ohri, 2014).

to solve this problem with easily accessible GUIs (Rödiger et al., 2012). However, the work on this additions has been recently started and is still in progress.

Acknowledgment

Part of this work was funded by the BMBF InnoProfile-Transfer-Projekt 03 IPT 611X, the European Regional Development Fund (ERDF/EFRE) on behalf of the European Union, the Sächsische Aufbaubank (Free State of Saxony, Germany) and the Russian Ministry of Education and Science (project No. RFMEFI62114X0003), with usage of scientific equipment of Center for collective use 'Biotechnology' at All-Russia Research Institute of Agricultural Biotechnology. We would like to thank the R community. We would like to thank Mario Menschikowski (Technical University Dresden) for the droplet digital PCR samples.

Bibliography

M. G. Almiron, B. Lopes, A. L. C. Oliveira, A. C. Medeiros, and A. C. Frery. On the numerical accuracy of spreadsheets. *Journal of Statistical Software*, 34(4):1–29, 4 2010. URL `http://www.jstatsoft.org/v34/i04`. [p128]

R. Bååth. The state of naming conventions in R. *The R Journal*, 4(2):74–75, Dec. 2012. URL `http://journal.r-project.org/archive/2012-2/RJournal_2012-2_Baaaath.pdf`. [p129]

P. Biecek and M. Kosinski. *archivist: Tools for Storing, Restoring and Searching for R Objects*, 2015. URL `http://CRAN.R-project.org/package=archivist`. R package version 1.3. [p130]

K. A. Blagodatskikh, S. Roediger, and M. Burdukiewicz. *RDML: Importing Real-Time Thermo Cycler (qPCR) Data from RDML Format Files*, 2015. URL `http://CRAN.R-project.org/package=RDML`. R package version 0.8-4. [p128]

L. D. Brown, T. T. Cai, and A. Dasgupta. Interval estimation for a binomial proportion. *Statistical Science*, 16:101–133, 2001. [p140]

M. Burdukiewicz, S. Roediger, and B. Jacobs. *dpcR: Digital PCR Analysis*, 2015. URL `http://CRAN.R-project.org/package=dpcR`. R package version 0.1.4.0. [p127]

P. Burns. Spreadsheet addiction. online, 2014. URL `http://web.archive.org/web/20141009042532/http://www.burns-stat.com/documents/tutorials/spreadsheet-addiction/`. [p128]

S. A. Bustin. The reproducibility of biomedical research: Sleepers awake! *Biomolecular Detection and Quantification*, 2:35–42, Dec. 2014. [p127]

I. Castro-Conde and J. de Uña-Álvarez. sgof: An R package for multiple testing problems. *The R Journal*, 6(2):96–113, Dec. 2014. [p127]

W. Chang, J. Cheng, J. J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2015. URL http://CRAN.R-project.org/package=shiny. R package version 0.12.1. [p31, 140]

P. Corbisier, L. Pinheiro, S. Mazoua, A.-M. Kortekaas, P. Y. J. Chung, T. Gerganova, G. Roebben, H. Emons, and K. Emslie. DNA copy number concentration measured by digital and droplet digital quantitative PCR using certified reference materials. *Analytical and Bioanalytical Chemistry*, 407(7): 1831–1840, 2015. [p140, 141, 143]

A. J. Durán, M. Pérez, and J. L. Varona. The misfortunes of a trio of mathematicians using computer algebra systems. Can we trust in them? *Notices of the American Mathematical Society*, 2014. [p128]

H. Dvinge and P. Bertone. HTqPCR: High-throughput analysis and visualization of quantitative real-time PCR data in R. *Bioinformatics*, 25(24):3325–3326, Dec. 2009. [p128]

C. Ekstrom, I. M. Skovgaard, and T. Martinussen. *kulife: Datasets and Functions from the (Now Non-Existing) Faculty of Life Sciences, University of Copenhagen*, 2013. URL http://CRAN.R-project.org/package=kulife. R package version 0.1-14. [p128]

M. P. Fay. Two-sided exact tests and matching confidence intervals for discrete data. *The R Journal*, 2 (1):53–58, June 2010. [p144]

C. Gandrud. *Reproducible Research with R and RStudio*. Chapman and Hall/CRC, July 2013. [p129]

R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. H. Yang, and J. Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5(10):R80, 2004. [p128, 129]

M. Guescini, D. Sisti, M. B. L. Rocchi, L. Stocchi, and V. Stocchi. A new real-time PCR method to overcome significant quantitative inaccuracy due to slight amplification inhibition. *BMC Bioinformatics*, 9(326), July 2008. [p130]

H. Hofmann, A. Unwin, and D. Cook. Let graphics tell the story – Datasets in R. *The R Journal*, 5(1): 117–130, June 2013. [p129]

J. Huggett, J. O'Grady, and S. Bustin. How to make mathematics biology's next and better microscope. *Biomolecular Detection and Quantification*, 1(1):A1–A3, 2014. [p127]

J. F. Huggett, C. A. Foy, V. Benes, K. Emslie, J. A. Garson, R. Haynes, J. Hellemans, M. Kubista, R. D. Mueller, T. Nolan, M. W. Pfaffl, G. L. Shipley, J. Vandesompele, C. T. Wittwer, and S. A. Bustin. The digital MIQE guidelines: Minimum information for publication of quantitative digital PCR experiments. *Clinical Chemistry*, 59(6):892–902, June 2013. [p127, 140]

J. F. Huggett, S. Cowen, and C. A. Foy. Considerations for digital PCR as an accurate molecular diagnostic tool. *Clinical Chemistry*, 61(1):79–88, Oct. 2015a. [p140]

J. F. Huggett, J. O'Grady, and S. Bustin. qPCR, dPCR, NGS — A journey. *Biomolecular Detection and Quantification*, 3:A1–A5, Mar. 2015b. [p127]

M. Jahn, J. Seifert, M. von Bergen, A. Schmid, B. Bühler, and S. Müller. Subpopulation-proteomics in prokaryotic populations. *Current Opinion in Biotechnology*, 24(1):79–87, Feb. 2013. [p141]

M. Jahn, C. Vorpahl, D. Türkowsky, M. Lindmeyer, B. Bühler, H. Harms, and S. Müller. Accurate determination of pasmid copy number of flow-sorted cells using droplet digital PCR. *Analytical Chemistry*, 86(12):5969–5976, June 2014. [p141]

D. A. Khodakov and A. V. Ellis. Recent developments in nucleic acid identification using solid-phase enzymatic assays. *Microchimica Acta*, 181(13–14):1633–1646, 2014. [p127]

J. D. Kloke and J. W. McKean. Rfit: Rank-based estimation for linear models. *The R Journal*, 4(2):57–64, Dec. 2012. [p139]

M. Kohl. *SLqPCR: Functions for Analysis of Real-Time Quantitative PCR Data at SIRS-Lab GmbH*. SIRS-Lab GmbH, Jena, 2007. URL www.sirs-lab.com. [p128]

M. Kuhn. CRAN task view: Reproducible research, 2015. URL http://CRAN.R-project.org/view=ReproducibleResearch. Version 2015-06-18. [p129]

T. J. Leeper. Archiving reproducible research and dataverse with R. *The R Journal*, 6(1):151–158, 2014. [p129]

S. Lefever, J. Hellemans, F. Pattyn, D. R. Przybylski, C. Taylor, R. Geurts, A. Untergasser, J. Vandesompele, and RDML Consortium. RDML: Structured language and reporting guidelines for real-time quantitative PCR data. *Nucleic Acids Research*, 37(7):2065–2069, Apr. 2009. [p128, 135]

Z. Liu and S. Pounds. An R package that automatically collects and archives details for reproducible computing. *BMC Bioinformatics*, 15(1):138, May 2014. [p129]

T. Luo, L. Jiang, W. Sun, G. Fu, J. Mei, and Q. Gao. Multiplex real-time PCR melting curve assay to detect drug-resistant mutations of mycobacterium tuberculosis. *Journal of Clinical Microbiology*, 49 (9):3132–3138, Sept. 2011. [p133]

N. Majumdar, T. Wessel, and J. Marks. Digital PCR modeling for maximal sensitivity, dynamic range and measurement precision. *PLoS ONE*, 10(3):e0118833, Mar. 2015. [p140]

J. Mar. *qpcrNorm: Data-Driven Normalization Strategies for High-Throughput qPCR Data*, 2009. URL `http://bioconductor.org`. R package version 1.26.0. [p128]

M. V. Matz. *MCMC.qpcr: Bayesian Analysis of qRT-PCR Data*, 2015. URL `http://CRAN.R-project.org/package=MCMC.qpcr`. R package version 1.2. [p128]

M. N. McCall1, H. R. McMurray, H. Land, and A. Almudevar. On non-detects in qPCR data. *Bioinformatics*, 30(16):2310–2316, Aug. 2014. [p128]

B. D. McCullough and D. A. Heiser. On the accuracy of statistical procedures in Microsoft Excel 2007. *Computational Statistics & Data Analysis*, 52(10):4570–4578, June 2008. [p128]

P. Michna. *RNetCDF: Interface to NetCDF Datasets*, 2015. URL `http://CRAN.R-project.org/package=RNetCDF`. R package version 1.7-3; with contributions from Milton Woods. [p129]

P. Michna and M. Woods. RNetCDF – A package for reading and writing NetCDF datasets. *The R Journal*, 5(2):29–37, Dec. 2013. [p129]

C. A. Milbury, Q. Zhong, J. Lin, M. Williams, J. Olson, D. R. Link, and B. Hutchison. Determining lower limits of detection of digital PCR assays for cancer-related gene mutations. *Biomolecular Detection and Quantification*, 1(1):8–22, Sept. 2014. [p127, 140]

A. A. Morley. Digital PCR: A brief history. *Biomolecular Detection and Quantification*, 1(1):1–2, Sept. 2014. [p127]

P. Murrell. It's not what you draw, it's what you don't draw. *The R Journal*, 4(2):13–18, Dec. 2012. [p128, 129]

P. Murrell. The gridGraphics package. *The R Journal*, 7(1):152–163, June 2015. [p128]

J. D. Neve and J. Meys. *unifiedWMWqPCR: Unified Wilcoxon-Mann-Whitney Test for qPCR Data*, 2014. URL `http://bioconductor.org`. R package version 1.4.0. [p128]

J. D. Neve, J. Meys, J.-P. Ottoy, L. Clement, and O. Thas. unifiedWMWqPCR: The unified Wilcoxon-Mann-Whitney test for analyzing RT-qPCR data in R. *Bioinformatics*, 30(17):2494–2495, 2014. [p128]

G. J. Nixon, H. F. Svenstrup, C. E. Donald, C. Carder, J. M. Stephenson, S. Morris-Jones, J. F. Huggett, and C. A. Foy. A novel approach for evaluating the performance of real time quantitative loop-mediated isothermal amplification-based methods. *Biomolecular Detection and Quantification*, 2:4–10, Dec. 2014. [p127]

D. Nolan and D. L. Temple. *XML and Web Technologies for Data Sciences with R*. Springer-Verlag, New York, 2014. [p146]

J. Oh. Automatic conversion of tables to LongForm dataframes. *The R Journal*, 6(2):16–26, 2014. [p129]

A. Ohri. *R for Cloud Computing – An Approach for Data Scientists*. Springer-Verlag, New York, 2014. [p146]

J. Ooms. Directions for improved dependency versioning in R. *The R Journal*, 5(1):197–207, June 2013. [p129]

S. Pabinger, S. Rödiger, A. Kriegner, K. Vierlinger, and A. Weinhäusel. A survey of tools for the analysis of quantitative PCR (qPCR) data. *Biomolecular Detection and Quantification*, 1, 2014. [p127, 128, 140]

Y. Pan, X. Yan, and J. Li. *qPCR.CT: qPCR Data Analysis and Plot Package*, 2012. URL http://CRAN.R-project.org/package=qPCR.CT. R package version 1.1. [p128]

S. L. Pape. *EasyqpcR: EasyqpcR for Easy Analysis of Real-Time PCR Data at IRTOMIT-INSERM U1082*. IRTOMIT-INSERM U1082, 2012. URL http://irtomit.labo.univ-poitiers.fr/. [p128]

J. R. Perkins, J. M. Dawes, C. Orengo, S. B. McMahon, D. L. Bennett, and M. Kohl. ReadqPCR and NormqPCR: R packages for the reading, quality checking and normalisation of RT-qPCR quantification cycle (Cq) data. *BMC Genomics*, 13(296), 2012. [p128]

R Core Team. *R Data Import/Export*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL http://www.R-project.org/. [p129]

C. Ritz and A.-N. Spiess. qpcR: an R package for sigmoidal model selection in quantitative real-time polymerase chain reaction analysis. *Bioinformatics*, 24(13):1549–1551, July 2008. PMID: 18482995. [p130]

S. Rödiger, T. Friedrichsmeier, P. Kapat, and M. Michalke. RKWard: A comprehensive graphical user interface and integrated development environment for statistical analysis with R. *Journal of Statistical Software*, 49(9):1–34, 2012. URL http://www.jstatsoft.org/v49/i09. [p129, 146]

S. Rödiger, A. Böhm, and I. Schimke. Surface melting curve analysis with R. *The R Journal*, 5(2):37–53, Dec. 2013a. [p128, 130, 133, 136, 138]

S. Rödiger, P. Schierack, A. Böhm, J. Nitschke, I. Berger, U. Frömmel, C. Schmidt, M. Ruhland, I. Schimke, D. Roggenbuck, W. Lehmann, and C. Schröder. A highly versatile microscope imaging technology platform for the multiplex real-time detection of biomolecules and autoimmune antibodies. *Advances in Biochemical Engineering/Biotechnology*, 133:35–74, 2013b. [p127, 128, 138]

S. Rödiger, C. Liebsch, C. Schmidt, W. Lehmann, U. Resch-Genger, U. Schedler, and P. Schierack. Nucleic acid detection based on the use of microbeads: A review. *Microchimica Acta*, 181(11–12): 1151–1168, 2014. [p127, 138]

S. Rödiger, M. Burdukiewicz, and P. Schierack. chipPCR: an R package to pre-process raw data of amplification curves. *Bioinformatics*, in press. doi: 10.1093/bioinformatics/btv205. [p130, 131, 139]

S. Roediger. *MBmca: Nucleic Acid Melting Curve Analysis on Microbead Surfaces with R*, 2015. URL http://CRAN.R-project.org/package=MBmca. R package version 0.0.3-5. [p128]

S. Roediger and M. Burdukiewicz. *chipPCR: Toolkit of Helper Functions to Pre-Process Amplification Data*, 2014. URL http://CRAN.R-project.org/package=chipPCR. R package version 0.0.8-4. [p128]

RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA, 2012. URL http://www.rstudio.com/. [p129, 160]

J. M. Ruijter, M. W. Pfaffl, S. Zhao, A. N. Spiess, G. Boggy, J. Blom, R. G. Rutledge, D. Sisti, A. Lievens, K. De Preter, S. Derveaux, J. Hellemans, and J. Vandesompele. Evaluation of qPCR curve analysis methods for reliable biomarker discovery: bias, resolution, precision, and implications. *Methods*, 59 (1):32–46, Jan. 2013. [p127, 128, 130, 133]

J. M. Ruijter, P. Lorenz, J. M. Tuomi, M. Hecker, and M. J. B. v. d. Hoff. Fluorescent-increase kinetics of different fluorescent reporters used for qPCR depend on monitoring chemistry, targeted sequence, type of DNA input and PCR efficiency. *Microchimica Acta*, 181(13–14):1689–1696, Oct. 2014. [p127]

D. A. Selck, M. A. Karymov, B. Sun, and R. F. Ismagilov. Increased robustness of single-molecule counting with microfluidics, digital isothermal amplification, and a mobile phone versus real-time kinetic measurements. *Analytical Chemistry*, 85(22):11129–11136, Nov. 2013. [p127]

A.-N. Spiess. *qpcR: Modelling and Analysis of Real-Time PCR Data*, 2014. URL http://CRAN.R-project.org/package=qpcR. R package version 1.4-0. [p128]

A.-N. Spiess, C. Deutschmann, M. Burdukiewicz, R. Himmelreich, K. Klat, P. Schierack, and S. Rödiger. Impact of smoothing on parameter estimation in quantitative DNA amplification. *Clinical Chemistry*, 61(2):379–388, 2015. [p128, 138]

D. Svec, A. Tichopad, V. Novosadova, M. W. Pfaffl, and M. Kubista. How good is a PCR efficiency estimate: Recommendations for precise and robust qPCR efficiency assessments. *Biomolecular Detection and Quantification*, 3:9–16, Mar. 2015. [p130]

D. A. Swan. *DivMelt: HRM Diversity Assay Analysis Tool*, 2013. URL http://CRAN.R-project.org/package=DivMelt. R package version 1.0.3; with contributions from Craig A Magaret and Matthew M Cousins. [p128]

P. M. Valero-Mora and R. Ledesma. Graphical user interfaces for R. *Journal of Statistical Software*, 49(1): 1–8, June 2012. URL http://www.jstatsoft.org/v49/i01. [p129]

M. van der Loo. *settings: Software Option Settings Manager for R*, 2015. URL http://CRAN.R-project.org/package=settings. R package version 0.2.2. [p130]

E. Viturro, C. Altenhofer, B. Zölch, A. Burgmaier, I. Riedmaier, and M. W. Pfaffl. Microfluidic high-throughput reverse-transcription quantitative PCR analysis of liver gene expression in lactating animals. *Microchimica Acta*, 181(13-14):1725–1732, Oct. 2014. [p127]

W. Wang and G. Shan. ExactCIdiff: An R package for computing exact confidence intervals for the difference of two proportions. *The R Journal*, 5(2):62–71, Dec. 2013. [p144]

J. Wu, R. Kodzius, W. Cao, and W. Wen. Extraction, amplification and detection of DNA in microfluidic chip-based assays. *Microchimica Acta*, 181(13-14):1611–1631, Oct. 2014. [p127]

M. Zeller, W.-C. L. A. Guazzelli, and G. Williams. PMML: An open standard for sharing models. *The R Journal*, 1(1):60–65, June 2009. [p129]

J. D. Zhang, R. Biczok, and M. Ruschhaupt. *ddCt: The ddCt Algorithm for the Analysis of Quantitative Real-Time PCR (qRT-PCR)*, 2015. URL http://bioconductor.org. R package version 1.22.0. [p128]

Stefan Rödiger (corresponding author)
Faculty of Natural Sciences
Brandenburg University of Technology Cottbus–Senftenberg
Senftenberg
Germany
Stefan.Roediger@b-tu.de

Michał Burdukiewicz
University of Wroclaw
Faculty of Biotechnoloy
Department of Genomics
Wroclaw
Poland
michalburdukiewicz@gmail.com

Konstantin Blagodatskikh
All-Russia Research Institute of Agricultural Biotechnology
Center for collective use 'Biotechnology'
Moscow
Russia
k.blag@yandex.ru

Michael Jahn
Helmholtz Centre for Environmental Research - UFZ
Flow cytometry group / Environmental microbiology
Leipzig
Germany
michael.jahn@ufz.de

Peter Schierack
Faculty of Natural Sciences
Brandenburg University of Technology Cottbus–Senftenberg
Senftenberg
Germany
Peter.Schierack@hs-lausitz.de

The gridGraphics Package

by Paul Murrell

Abstract The **gridGraphics** package provides a function, `grid.echo()`, that can be used to convert a plot drawn with the **graphics** package to a visually identical plot drawn using **grid**. This conversion provides access to a variety of **grid** tools for making customisations and additions to the plot that are not possible with the **graphics** package.

Introduction

The core graphics system in R is divided into two main branches, one based on the **graphics** package and one based on the **grid** package, with many other packages building on top of one or other of these graphics systems (see Figure 1).

The **graphics** package is older and provides an emulation of the original GRZ graphics system from S (Becker and Chambers, 1984). The newer **grid** package, although its performance is actually slower, provides greater flexibility and additional features compared to the **graphics** package. In particular, a plot drawn with **grid** can be manipulated and edited in many more ways than a plot drawn with the **graphics** package.

This article describes a new package, called **gridGraphics**, that allows a plot drawn with **graphics** to be converted into an identical plot drawn with **grid**, thereby allowing the plot to be manipulated using all of the tools available in **grid**.

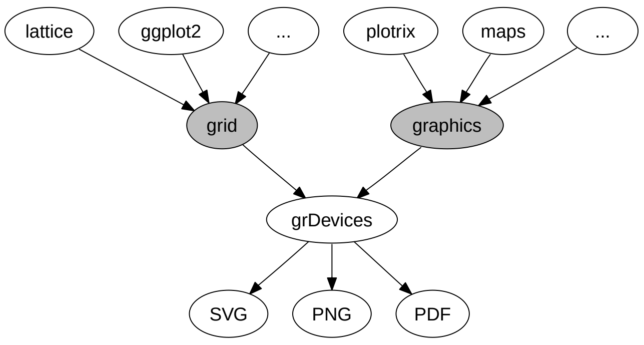


Figure 1: The structure of the core graphics system in R. The **lattice** package (Sarkar, 2008), the **ggplot2** package (Wickham, 2009) and many others are built on top of the **grid** package; the **plotrix** package (Lemon, 2006), the **maps** package (Brownrigg, 2013) and *many* others are built on top of the **graphics** package.

The `grid.echo()` function

The **gridGraphics** package provides a single main function called `grid.echo()`. By default, this function takes whatever has been drawn by the **graphics** package on the current graphics device and redraws it using **grid**. The following code provides a simple demonstration. We first draw a scatterplot using `plot()` from the **graphics** package, then we call `grid.echo()` to replicate the plot with **grid**. Figure 2 shows that the original plot and the replicated plot are identical.

```
> plot(mpg ~ disp, mtcars, pch = 16)
> library(gridGraphics)
> grid.echo()
```

The following sections will attempt to demonstrate why, although the plots appear identical to the eye, there are important advantages that arise from using **grid** to do the drawing.

Manipulating grobs

One advantage of drawing the plot with **grid** is that there is an object, a **grid** *grob*, recorded for each separate component of the plot that we have drawn. We can see that list of grobs with a call

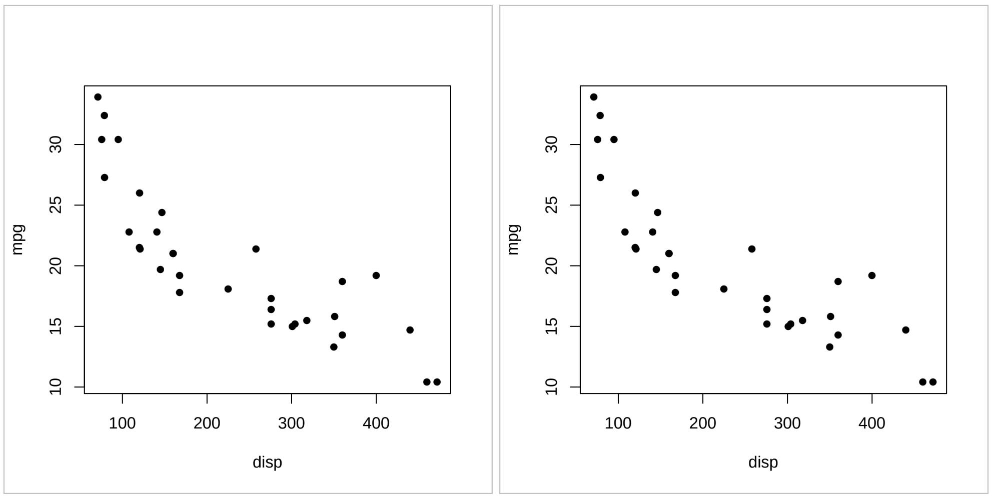


Figure 2: On the left, a scatterplot drawn with the **graphics** package and, on the right, the result of `grid.echo()`, which produces the same scatterplot using the **grid** package.

to the `grid.ls()` function, as shown below. There is a grob called `graphics-plot-1-points-1` that represents the data symbols in the plot, there is a grob called `graphics-plot-1-xlab-1` that represents the x-axis label, and so on.

```
> grid.ls()
graphics-plot-1-points-1
graphics-plot-1-bottom-axis-line-1
graphics-plot-1-bottom-axis-ticks-1
graphics-plot-1-bottom-axis-labels-1
graphics-plot-1-left-axis-line-1
graphics-plot-1-left-axis-ticks-1
graphics-plot-1-left-axis-labels-1
graphics-plot-1-box-1
graphics-plot-1-xlab-1
graphics-plot-1-ylab-1
```

The **grid** package provides several functions to manipulate these grobs. For example, the code below uses the `grid.edit()` function to rotate the tick labels on the x-axis of the plot to 45 degrees (and turns them red so that the change is easy to spot; see Figure 3). This is a simple example of a customisation that is impossible or very difficult in the **graphics** package, but is quite straightforward once the plot has been converted to **grid**.

```
> grid.edit("graphics-plot-1-bottom-axis-labels-1", rot = 45,
+           gp = gpar(col = "red"))
```

To provide a more sophisticated example, consider the conditioning plot produced by the following code with a call to the `coplot()` function from the **graphics** package (see the left-hand plot in Figure 4).

```
> coplot(lat ~ long | depth, quakes, pch = 16, cex = .5,
+        given.values = rbind(c(0, 400), c(300, 700)))
```

This is an example of a much more complex plot with many different components. Functions that produce this sort of complex plot can struggle to provide arguments to fine tune all possible elements of the plot. For example, suppose that we want to modify the "conditioning panel" at the top of the plot so that the background is a solid colour and the bars are filled in white. This specific task is probably not a common one for most people, but the point of this example is to represent a class of problems where a small detail within a complex plot needs to be modified.

The `coplot()` function does have an argument `bar.bg` to control the fill colour for the bars, as demonstrated in the code below (see the right-hand plot in Figure 4). However, there is no argument that allows us to control the background colour for the panel behind the bars.

```
> coplot(lat ~ long | depth, quakes, pch = 16, cex = .5,
+        given.values = rbind(c(0, 400), c(300, 700)),
+        bar.bg = c(num = "white"))
```

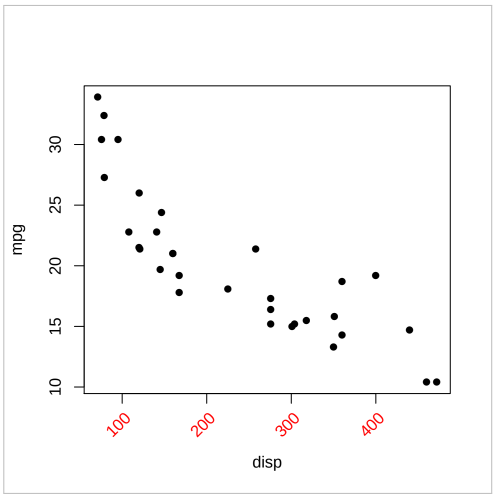


Figure 3: A scatterplot that was drawn using `plot()` from **graphics**, then redrawn using **grid** via the `grid.echo()` function, then edited using `grid.edit()`.

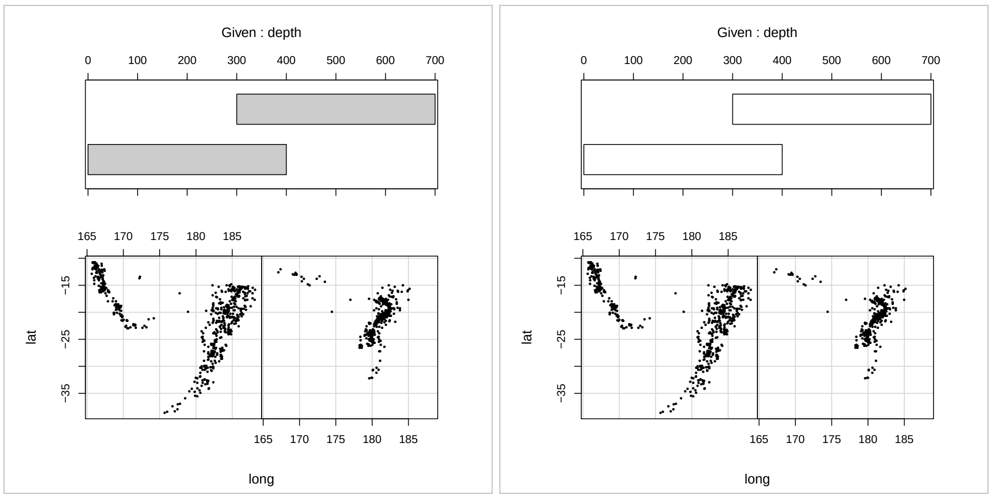


Figure 4: A conditioning plot produced by the `coplot()` function from the **graphics** package. The right-hand version of the plot demonstrates the use of the `bar.bg` argument to customise the fill colour of the bars in the conditioning panel at the top of the plot.

If we replicate this plot using **grid**, we have more tools available to be able to manipulate the plot. The `grid.echo()` function can replicate this plot and gives an identical result to that shown in Figure 4.

The call to `grid.echo()` shown below also demonstrates the use of the `prefix` argument, which can be used to control the naming of the grobs that `grid.echo()` draws. The grobs created by this call to `grid.echo()` all have names that start with `"cp"` rather than the default `"graphics"` prefix.

```
> grid.echo(prefix = "cp")
```

Once this conversion has taken place, we now have **grid** grobs that represent all components of the plot. In particular, there is a grob called `"cp-plot-4-box-1"` that draws the border around the conditioning panel. We can edit that grob to give it a fill colour (see the left-hand plot in Figure 5).

```
> grid.edit("cp-plot-4-box-1", gp = gpar(fill = "red"))
```

This provides another demonstration that converting a plot to **grid** provides access to all of the components of the plot, which allows fine control over details of the plot that cannot be controlled via the arguments to the original high-level function that created the plot.

In this particular example, we have also created a new problem, because the border on the conditioning plot is drawn *after* the bars, so the bars are now obscured. Fortunately, we can fix this as well with further tools that **grid** provides for manipulating grobs.

In the following code, we call the `grid.grab()` function to create a single grob (a *gTree*) that contains all of the other grobs on the page. We then call the `grid.reorder()` function to change the order of the grobs within the gTree. The code specifies that the border grob will be drawn first (behind all other components in the plot). Finally, we redraw the reordered plot with the `grid.draw()` function to get the final result that we were after (see the right-hand plot in Figure 5).

```
> gt <- grid.grab()
> gt <- reorderGrob(gt, "cp-plot-4-box-1")
> grid.newpage()
> grid.draw(gt)
```

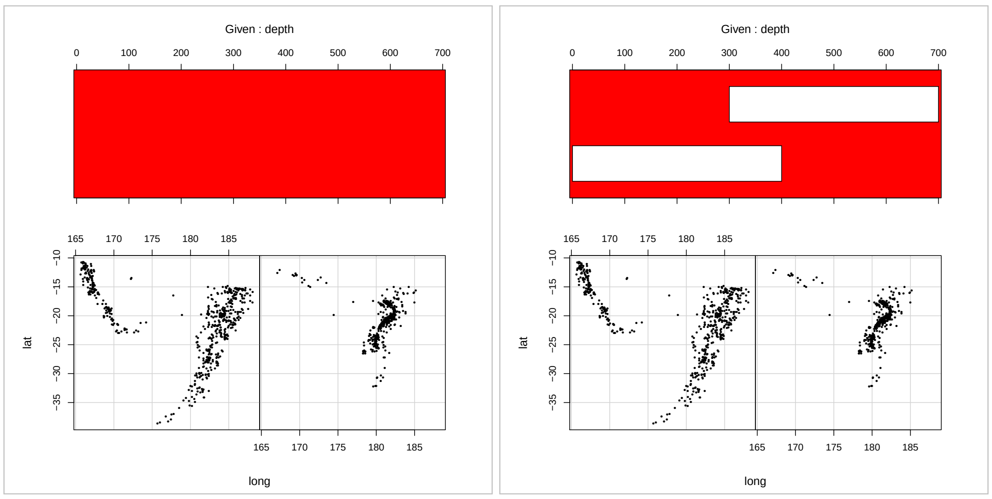


Figure 5: On the left, the plot from Figure 4 after conversion with `grid.echo()`, followed by editing to fill the rectangle that draws a border around the top conditioning panel (in red so that the change is visible). On the right, the edited plot has been reordered so that the border around the conditioning panel is drawn first (behind everything else).

This access to individual components of a plot and the ability to manipulate those components is one benefit of converting a **graphics** plot to **grid**.

Making use of viewports

Another advantage of using **grid** is that we can make use of *viewports*. Viewports are similar to the different plotting regions that the **graphics** package uses for drawing (see Figure 6), but in a **grid** plot there can be an unlimited number of viewports and all viewports are accessible at any time. In the **graphics** package there is only the current plot region, figure margins, and outer margins to work with.

Figure 7 shows a diagram of the hierarchy of viewports that `grid.echo()` created when we replicated the simple scatterplot in Figure 2. This shows that **gridGraphics** produces quite a lot of viewports (even for a simple plot), but there is a coherent structure to the viewports, so the complexity can be navigated without too much difficulty.

In general, the names of the viewports reflect the plot regions that they mimic in the original plot. At the top of the hierarchy of viewports is a viewport called `ROOT`, which represents the entire page (this viewport is always present). Below that is a viewport called `graphics-root` and that represents the area of the page that `grid.echo()` has drawn into (by default, also the whole page). The next viewport down is called `graphics-inner` and this represents the region that is the whole page minus the outer margins. Below that are two viewports, `graphics-figure-1` and `graphics-figure-1-clip`, both of which correspond to the figure region (the grey area in Figure 6). There are two viewports because one has clipping turned on, so that drawing within that region cannot extend beyond the boundaries of the region, and one has clipping turned off. Below each of the figure region viewports are one or more viewports representing the plot region, called either `graphics-plot-1` or `graphics-plot-1-clip`. Again, the difference between these two plot viewports is whether clipping is on or off. By having the `graphics-plot-1` viewport beneath both figure viewports, we can represent all possible values of the `par("xpd")` settings: clipping to the figure region, or clipping to the plot region, or no clipping at all. The bottom layer of viewports represent the plot region again, but this time with a viewport that has

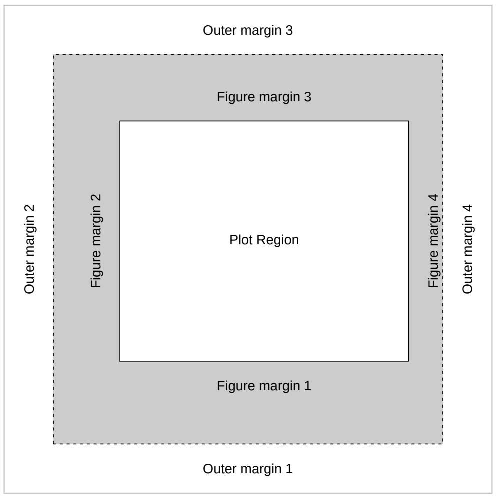


Figure 6: The plotting regions involved in the drawing of a **graphics** plot.

scales to represent the plot axes. The reason for this additional layer is so that we can reproduce plots that make use of more than one set of axes (e.g., two different y-axis scales).

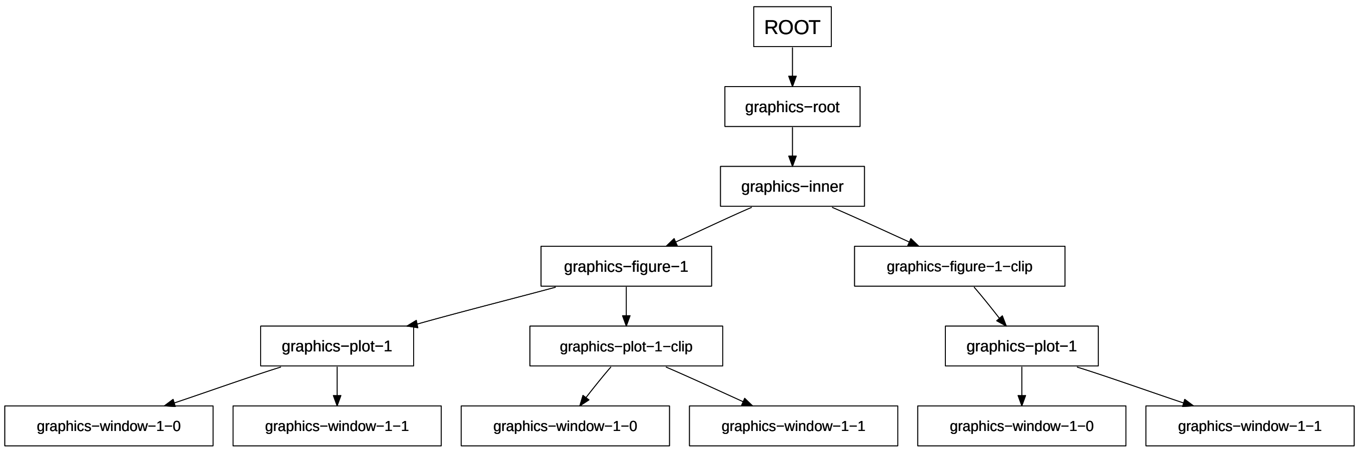


Figure 7: A diagram of the viewports that were created by the `grid.echo()` function when it drew the plot in Figure 2.

The upside to having so many viewports is that the **grid** package provides functions to navigate between viewports. So we can have a lot of viewports on the page at once, but switch between them if we want to add drawing within different viewports. As an example, the following code uses the `downViewport()` function to revisit the plot region viewport that was created by `grid.echo()` and draws a red rectangle around the border (see Figure 8). The `upViewport()` function is then used to take us back to the whole page (the `ROOT` viewport).

```
> downViewport("graphics-plot-1")
> grid.rect(gp = gpar(col = "red", lwd = 3))
> upViewport(0)
```

Once again, a more sophisticated demonstration can be provided if we consider the more complex conditioning plot from Figure 4. Another limitation of the `coplot()` function, because it is based on the **graphics** package, is that there is no way to add further drawing to the conditioning panel at the top of the plot.

This plot has several different panels so the replication created by `grid.echo()` generates many different viewports, including viewports used to draw the conditioning panel at the top of the plot.

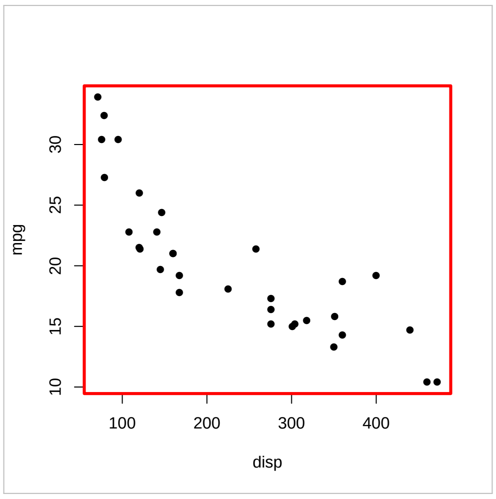


Figure 8: The echoed plot from Figure 2 with a rectangle added by revisiting the viewport that corresponds to the plot region.

With **grid**, all of these viewports can be revisited after the plot has been drawn. In the following code, we revisit the viewport used to draw the conditioning panel and draw some grid lines in it.

Those new lines will be drawn on top of everything else, so additional manipulations, similar to the reordering performed for Figure 5, can also be carried out to push the segments behind everything else (code not shown). The result is shown in Figure 9.

```
> downViewport("cp-window-4-1")
> v <- unit(seq(0, 700, 100), "native")
> grid.segments(v, 0, v, 1, gp = gpar(col = "red"), name = "grid")
> upViewport(0)
```

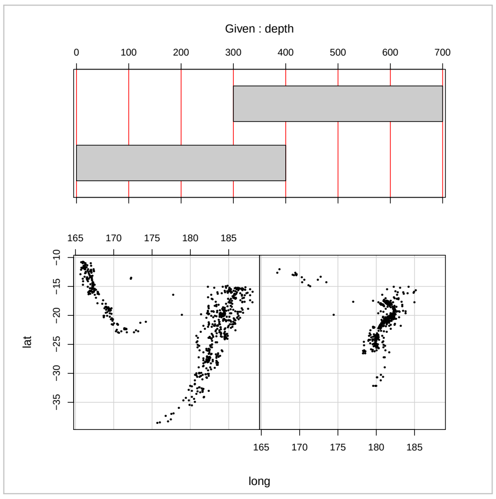


Figure 9: The conditioning plot from Figure 4 with a reference grid added (in red so that the change is visible) to the conditioning panel at the top of the plot.

Another limitation of the original `coplot()` function is that it insists on occupying the entire page. Another advantage of working with **grid** grobs and viewports is that they can be nested within each other to any level. This means that once the output from `coplot()` has been replicated as **grid** output, it can be drawn within a **grid** viewport and combined on a page with other plots.

The following code creates a **grid** viewport occupying the bottom 70% of the page and then replicates the conditioning plot only using that part of the page. This code demonstrates another way to call the `grid.echo()` function. Rather than calling the **graphics** function `coplot()` to draw a plot and then calling `grid.echo()` to replicate it, we can define a function (with no arguments) that draws the plot and then provide that function as the first argument to `grid.echo()`. This way the plot is only drawn once, using **grid**. We also specify `newpage = FALSE` in the call to `grid.echo()` so that it just draws in the current viewport rather than starting a new page.

```
> cpfun <- function() {
+     coplot(lat ~ long | depth, quakes, pch = 16, cex = .5,
+            given.values = rbind(c(0, 400), c(300, 700)))
+ }
> grid.newpage()
> pushViewport(viewport(y = 0, height = .7, just = "bottom"))
> grid.echo(cpfun, newpage = FALSE, prefix = "cp")
> upViewport()
```

The next piece of code draws a **ggplot2** histogram in the top third of the page, so not only do we have a conditioning plot combined with another plot on the same page (something that was not at all possible with the original **graphics**-based conditioning plot), but we have a mixture of **graphics**-based and **grid**-based output on the same page (see Figure 10).

```
> library(ggplot2)
> pushViewport(viewport(y = 1, height = .33, just = "top"))
> gg <- ggplot(quakes) + geom_bar(aes(x = depth)) +
+       theme(axis.title.x = element_blank())
> print(gg, newpage = FALSE)
> upViewport()
```

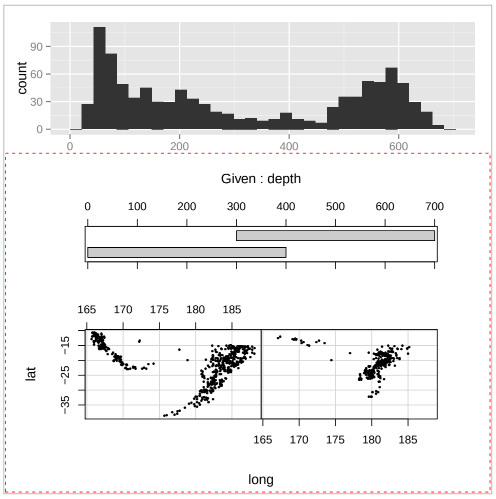


Figure 10: The conditioning plot from Figure 4 combined with a **ggplot2** histogram on the same page. A dashed red box has been drawn around the region that is occupied by the conditioning plot, to emphasise the fact that the conditioning plot does not occupy the entire page.

This sort of result—**grid**-based plots combined with **graphics**-based plots on the same page—can also be achieved using the **gridBase** package (Murrell, 2012). However, **gridBase** only allows plots from the two packages to coexist side-by-side on the same page; it does not provide any of the benefits of **grid** for **graphics**-based plots.

Exporting to SVG

Another benefit that we get from converting a **graphics** plot to **grid** is that the converted plot can then be exported to SVG via the **gridSVG** package (Murrell and Potter, 2014). This means that we gain the potential to add hyperlinks to the plot, animate components of the plot, add advanced SVG features to the plot, and add interactivity (possibly via JavaScript code).

As a simple example, the following code draws the scatterplot from Figure 2 with `plot()` from the **graphics** package, converts the plot to **grid** with `grid.echo()`, and then adds tooltips to each data symbol and exports the plot to SVG with the functions `grid.garnish()` and `grid.export()` from the **gridSVG** package. The SVG plot, as viewed in a browser, is shown in Figure 11.

```
> library(gridSVG)
> plot(mpg ~ disp, mtcars, pch = 16)
> grid.echo()
> grid.garnish("graphics-plot-1-points-1", group = FALSE, title = rownames(mtcars))
> grid.export("murrell-echo.svg")
```

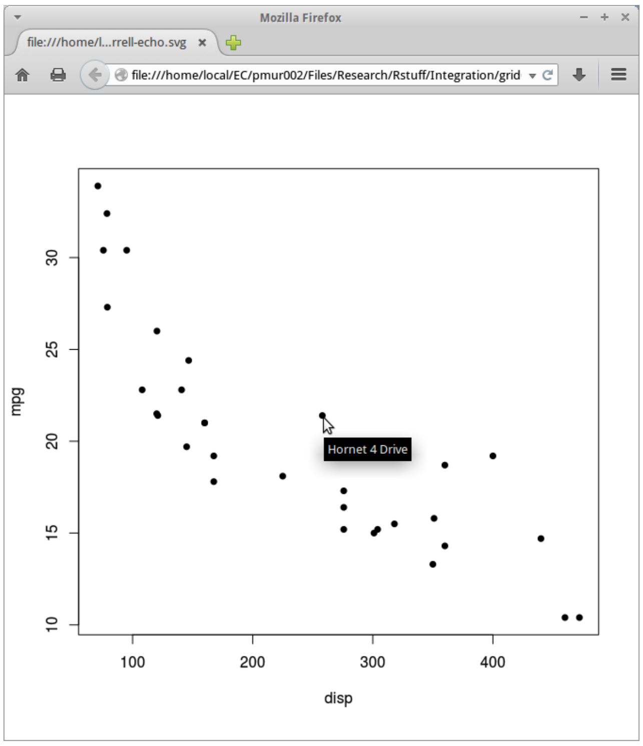

Mozilla Firefox
file:///home/l...rrell-echo.svg
file:///home/local/EC/pmur002/Files/Research/Rstuff/Integration/grid
mpg
10
15
20
25
30
100
200
300
400
disp
Hornet 4 Drive

Figure 11: The scatterplot plot from Figure 2, after conversion to **grid** using `grid.echo()`, exported to SVG, with tooltips added, using functions from the **gridSVG** package. The tooltips in this example may only work with Firefox.

A **graphics** plot that is converted by **gridGraphics** and then exported by **gridSVG** also has potential benefits for accessibility because, for example, text labels are exported as `<text>` elements, which can be recognised by screen reading software. This is not the case with the standard `svg()` graphics device, which exports text as `<path>` elements.

Naming schemes

Something that has been ignored in the examples so far is the fact that all manipulations of **grid** grobs, as well as navigation between **grid** viewports, rely on being able to identify, by name, which grob we want to manipulate or which viewport we want to visit. This section describes the naming scheme that the **gridGraphics** package uses to assign names to the grobs and viewports that it creates.

The names of grobs have the following basic pattern:

```
<prefix>-plot-<i>-<label>-<j>
```

where `<prefix>` is `graphics` by default, but can be specified in the call to `grid.echo()`, `<i>` reflects which plot the grob was drawn in (because there can be more than one plot region on the page), `<label>` reflects what sort of shape was drawn, and `<j>` is a numeric index that automatically increments when more than one of the same shape is drawn within the same plot. The full set of possible shape labels is given in Table 1.

The names of viewports have one of the following basic patterns:

```
<prefix>-root
<prefix>-inner<a>
<prefix>-figure-<i><a>
<prefix>-plot-<i><a>
<prefix>-window-<i><a>-<j><b>
```

where `<prefix>` is the same as for grob names. There is only ever one `root` viewport, which is parent to usually one `inner` viewport. Both `plot` and `window` viewports occupy the same region, but `window` viewports represent the axis scales. The `<i>` part of the name is a numeric index that is automatically incremented, each time that `plot.new()` is called. The `<j>` part is similar, but increments each time

Function	Description	<label>
plotXY()	Points, lines, etc. through data	points lines step Step spike brokenline
text()	Text in plot region	text
title()	Plot title, sub-title, and axis labels	main sub xlab ylab
axis()	Axes, including tick marks and labels	bottom-axis-line bottom-axis-ticks bottom-axis-labels (ditto left-, top-, and right-)
mtext()	Text in margins	mtext-bottom mtext-left mtext-top mtext-right mtext-<side>-outer
box()	Border rectangles for plot regions	box box-figure box-inner box-outer
segments()	Straight line segments	segments
arrows()	Segments with arrow heads	arrows
abline()	A straight line parameterised by slope and intercept, or horizontal or vertical constant	abline-ab abline-h abline-v
rect()	Rectangles	rect
polygon()	Polygons	polygon
path()	General path	path
rasterImage()	Raster image	raster
xspline()	X-Splines	xspline
clip()	Clipping region rectangle	clip
contour()	Contour lines	contour-<i>
image()	Filled rectangles	image-rect
symbols()	High-dimensional data symbols	symbols-circle symbols-square symbols-rect symbols-star symbols-thermo-box symbols-thermo-fill symbols-thermo-whisker-right symbols-thermo-whisker-left symbols-boxplot-box symbols-boxplot-lower-whisker symbols-boxplot-upper-whisker symbols-boxplot-median

Table 1: The labels used for different shapes (grobs) that can be created by grid.echo(). The "Function" column gives the name of the **graphics** function that produces the original shapes, the "Description" column describes what sort of shape is drawn, and the "<label>" column gives the names used for the **grid** grobs that the shapes are converted into.

that `plot.window()` is called. The <a> part of the name only occurs when `par()` is used to modify graphical parameters that affect the location of plot regions, for example, to modify the plot region via `par("pin")`. The <b> part is similar, but occurs when `par("usr")` is used to modify the axis scales.

For complex plots, like the conditioning plot in Figure 4, there can be a large number of grobs and viewports. To help with exploring the potentially large number of grobs and viewports, the **grid** package provides functions `grid.ls()`, `showGrob()`, and `showViewport()`, and the **gridDebug** package (Murrell and Ly, 2012) provides further tools.

Testing

The **gridGraphics** package is known to produce identical results for all examples in the **graphics** package help pages, plus the results of `demo("graphics")` (subject to the caveats described in the next section). However, it is still possible that there are some combinations of arguments in the low-level **graphics** functions that will not be emulated correctly by `grid.echo()`.

To test whether a **graphics**-based plot is reproduced correctly by `grid.echo()`, the **gridGraphics** package provides a `plotdiff()` function. This takes an expression as its first argument, which is assumed to be one or more calls to **graphics** functions. The `plotdiff()` function creates PDF and PNG files from evaluating the expression and from converting the result with `grid.echo()`, plus a PNG file showing differences between the **graphics** original and the **grid** copy, if there are any (a total of either four or five files).[1] A `plotdiffResult()` function is also provided to summarise the results of multiple calls to `plotdiff()`.

Limitations

There are a few **graphics** functions that `grid.echo()` cannot currently replicate: `persp()` for drawing 3-dimensional surfaces, `filled.contour()` for drawing a filled contour plot, and `recordGraphics()`, which allows delayed evaluation of drawing code.

For some other functions, there are a few details that do not reproduce exactly. For example, `grid.echo()` cannot reproduce text labels on contours drawn by `contour()` and `grid.echo()` will sometimes eliminate fewer axis tick labels than the `axis()` function.

More generally, `grid.echo()` cannot cope with code that opens or closes graphics devices or changes the current graphics device, it can only replicate a single page of plots, and if there is already a mixture of **grid** and **graphics** plots on a page, `grid.echo()` will only replicate the **graphics** plots.

An important situation where the above limitations will be encountered is within the RStudio IDE (RStudio, 2014). Plots drawn in the RStudio "Plots" pane will not reproduce well with `grid.echo()`. However, using a standard R graphics device with RStudio should still work.

Finally, the output from `grid.echo()` is only valid at the size it is first drawn. Subsequently resizing the graphics window or copying to another graphics device is likely to produce a distorted result.

Conclusion

The **gridGraphics** package provides a bridge between the **graphics** and **grid** packages (see Figure 12). The `grid.echo()` function converts a plot drawn using the **graphics** package into exactly the same result drawn using **grid**. In effect, the **gridGraphics** package provides an automated way to create a **grid**-based version of almost any plotting function that is based on the **graphics** package.

The benefit of converting to **grid** is that **grid** provides tools for making customisations and additions to a plot that are not possible with the **graphics** package:

- To allow customisation of fine details that are not accessible via the **graphics** package (e.g., Figures 3 and 5).
- To add extra drawing to a region of the plot that is inaccessible via the **graphics** package (e.g., Figure 9).
- To combine **graphics**-based plots with **grid**-based plots (e.g., Figure 10).

[1]On R versions prior to 3.2.0, and on systems where ImageMagick (ImageMagick Studio LLC, 2014) is not available, only the PDF files are created; there is no conversion or comparison. Prior to R 3.2.0, there will be some (typically very small) differences between some plots because of a bug in **grid**.

- To export a **graphics**-based plot to SVG with the **gridSVG** package (e.g., Figure 11).

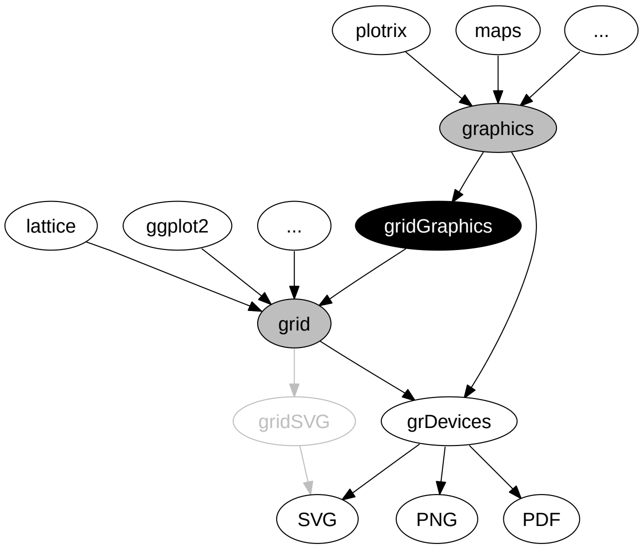


Figure 12: How the **gridGraphics** package fits into the structure of the core graphics system in R. The **gridSVG** package is also shown as an alternative route from **grid** to `SVG` output.

Availability

The **gridGraphics** package is available from CRAN, with ongoing development occurring on github.[2] Supporting material for this article, including a live version of Figure 11, is available from the following web site:

`https://www.stat.auckland.ac.nz/~paul/Reports/gridGraphics/`

Acknowledgements

Thanks to the anonymous reviewers who made several useful suggestions that improved this manuscript.

Bibliography

R. A. Becker and J. M. Chambers. *S: An Interactive Environment for Data Analysis and Graphics*. Wadsworth, Belmont, CA, 1984. [p151]

R. Brownrigg. *maps: Draw Geographical Maps*, 2013. URL `http://CRAN.R-project.org/package=maps`. R package version 2.3-2. [p151]

ImageMagick Studio LLC. *ImageMagick*, 2014. URL `http://www.imagemagick.org/`. [p160]

J. Lemon. plotrix: A package in the red light district of R. *R News*, 6(4):8–12, 2006. [p151]

P. Murrell. *gridBase: Integration of base and grid Graphics*, 2012. URL `http://CRAN.R-project.org/package=gridBase`. R package version 0.4-6. [p157]

P. Murrell and V. Ly. Debugging grid graphics. *The R Journal*, 4(2):19–27, Dec. 2012. URL `http://journal.r-project.org/archive/2012-2/RJournal_2012-2_Murrell+Ly.pdf`. [p160]

P. Murrell and S. Potter. The gridSVG package. *The R Journal*, 6(1):133–143, June 2014. URL `http://journal.r-project.org/archive/2014-1/murrell-potter.pdf`. [p157]

RStudio. *RStudio: Integrated Development Environment for R*, 2014. URL `http://www.rstudio.org/`. [p129, 160]

[2]`https://github.com/pmur002/gridgraphics`

D. Sarkar. *lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. URL http://lmdvr.r-forge.r-project.org. [p151]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer New York, 2009. URL http://had.co.nz/ggplot2/book. [p151]

Paul Murrell
Department of Statistics
The University of Auckland
New Zealand
paul@stat.auckland.ac.nz

fslr: Connecting the FSL Software with R

by John Muschelli, Elizabeth Sweeney, Martin Lindquist, and Ciprian Crainiceanu

Abstract We present the package **fslr**, a set of R functions that interface with FSL (FMRIB Software Library), a commonly-used open-source software package for processing and analyzing neuroimaging data. The **fslr** package performs operations on 'nifti' image objects in R using command-line functions from FSL, and returns R objects back to the user. **fslr** allows users to develop image processing and analysis pipelines based on FSL functionality while interfacing with the functionality provided by R. We present an example of the analysis of structural magnetic resonance images, which demonstrates how R users can leverage the functionality of FSL without switching to shell commands.

Glossary of acronyms

MRI	Magnetic Resonance Imaging/Image	FSL	FMRIB Software Library
PD	Proton Density	FAST	FMRIB's Automated Segmentation Tool
FLAIR	Fluid-Attenuated Inversion Recovery	FLIRT	FMRIB's Linear Image Registration Tool
MS	Multiple Sclerosis	BET	Brain Extraction Tool
FMRIB	Functional MRI of the Brain Group	FNIRT	FMRIB's Nonlinear Image Registration Tool
MNI	Montreal Neurological Institute		

Introduction

FSL (FMRIB Software Library) is a commonly-used software for processing and analyzing neuroimaging data (Jenkinson et al., 2012). This software provides open-source, command-line tools and a graphical user interface (GUI) for image processing tasks such as image smoothing, brain extraction (Smith, 2002), bias-field correction, segmentation (Zhang et al., 2001), and registration (Jenkinson and Smith, 2001; Jenkinson et al., 2002). Many of these functions are used extensively in medical imaging pipelines. According to a recent survey paper by Carp (2012), 13.9% of published neuroimaging studies used FSL.

There exist a number of R packages for reading and manipulating image data, including **AnalyzeFMRI** (Bordier et al., 2011), **RNiftyReg** (Clayden, 2015), and **fmri** (Tabelow and Polzehl, 2011) (see the Medical Imaging CRAN task view `http://CRAN.R-project.org/view=MedicalImaging` for more information). Although these packages are useful for performing image analysis, much of the fundamental functionality that FSL and other imaging software provide is not currently implemented in R. In particular, this includes algorithms for performing slice-time correction, motion correction, brain extraction, tissue-class segmentation, bias-field correction, co-registration, and normalization. This lack of functionality is currently hindering R users from performing complete analysis of image data within R. Instead of re-implementing FSL functions in R, we propose a user-friendly interface between R and FSL that preserves all the functionality of FSL, while retaining the advantages of using R. This will allow R users to implement complete imaging pipelines without necessarily learning software-specific syntax.

The **fslr** package relies heavily on the **oro.nifti** (Whitcher et al., 2011) package implementation of images (referred to as 'nifti' objects) that are in the Neuroimaging Informatics Technology Initiative (NIfTI) format, as well as other common image formats such as ANALYZE. **oro.nifti** also provides useful functions for plotting and manipulating images. **fslr** expands on the **oro.nifti** package by providing additional functions for manipulation of 'nifti' objects.

fslr workflow

The general workflow for most **fslr** functions that interface with FSL is as follows:

1. Filename or 'nifti' object is passed to **fslr** function.
2. FSL command is created within **fslr** function and executed using the `system` command.
3. Output is written to disk and/or read into R and returned from the function.

From the user's perspective, the input/output process is all within R. The advantage of this approach is that the user can read in an image, do manipulations of the 'nifti' object using standard syntax for arrays, and pass this object into the **fslr** function without using FSL-specific syntax written in a shell language. Also, one can perform image operations using FSL, perform operations on the 'nifti' object in R that would be more difficult using FSL, and then perform additional operations using FSL by passing that object to another **fslr** command. Thus, users can create complete pipelines for the analysis of imaging data by accessing FSL through **fslr**.

fslr setup

To use **fslr**, a working installation of FSL is required. The following code was run using FSL version 5.0.0. **fslr** must also have the path of FSL specified. If using R from a shell environment, and the `FSLDIR` environment variable is set (which can be done when installing FSL), **fslr** will use this as the path to FSL. If using R through a graphical user interface (GUI) such as RStudio (RStudio, Boston, MA), environmental variables and paths are not explicitly exported. Therefore, `FSLDIR` is not set, and the path to FSL can be specified using `options(fsl.path = "/path/to/fsl")`.

fslr also requires an output type for the format of images returned from FSL. Some **fslr** functions produce intermediate files that the user may want removed after the command is executed. When the filename argument is passed to a **fslr** command, the extension of the file does not need to be specified, but simply the prefix. When the command is executed, the FSL command appends an extension, and to remove this file, using the R command `file.remove`, the extension for the file is required. If working in a shell environment, **fslr** will use the environment variable for output type: `FSLOUTPUTTYPE`. If working in a GUI, the default is given by `NIFTI_GZ`, which returns compressed NIfTI images, ending in ".nii.gz". This can be changed by setting the `fsl.outputtype` option. See `http://fsl.fmrib.ox.ac.uk/fsl/fsl-4.1.9/fsl/formats.html` for a description of FSL output types.

The R code below is all that is needed to load the **fslr** package, set the path to FSL, and specify the output type for files, respectively.

```
library(fslr)
options(fsl.path = "/usr/local/fsl")
options(fsl.outputtype = "NIFTI_GZ")
```

Image preprocessing with fslr

We present a complete analysis of structural magnetic resonance imaging (MRI) data performed using **fslr** and R. Images were obtained from a patient with multiple sclerosis (MS) at 2 different visits (Sweeney et al., 2013), located at `bit.ly/FSL_Data`. At each visit, the image modalities obtained were T1-weighted (T1), T2-weighted (T2), fluid-attenuated inversion recovery (FLAIR), and proton density (PD). In this example we will perform a MRI bias-field correction using FAST (FMRIB's Automated Segmentation Tool; Zhang et al. 2001), co-register scans within visits to the T1 image of that visit, and register T1 images between visits. Once these operations have been performed, one can take within-modality difference images to see the changes between visits. We will also register all images to a common stereotaxic template, as this is common in population-based analyses.

Bias-field correction

MRI images typically exhibit good contrast between soft tissue classes, but intensity inhomogeneities in the radio frequency (RF) field can cause differences in the ranges of tissue types at different spatial locations. These inhomogeneities can cause problems with algorithms based on histograms, quantiles, or raw intensities (Zhang et al., 2001). Therefore, correction for image inhomogeneities is a crucial step in many analyses. FSL implements the bias-field correction from Guillemaud and Brady (1997) in its FAST segmentation pipeline (Zhang et al., 2001).

The `fsl_biascorrect` command from **fslr** will create the corrected images. We pass in the filename in the `file` argument, any additional options for FSL in the `opts` argument, such as `-v` for **v**erbose diagnostic outputs, and the **out**put **file**name in the `outfile` argument, which is our inhomogeneity-corrected image.

```
fsl_biascorrect(file = "01-Baseline_T1.nii.gz",
                outfile = "01-Baseline_T1_FSL_BiasCorrect",
                opts = "-v")
```

We can observe the difference in voxel values from the baseline T1 image compared to the bias-corrected version in Figure 1. In Panel A we display the T1 image, and in Panel B we display the bias-corrected T1 image. The T1 image is brighter in the middle of the image, while the bias-corrected image is more uniform in white matter (brighter regions). As this difference may be hard to distinguish visually, we present the scatterplot of these images in Figure 1C, using the **ggplot2** package (Wickham, 2009). Note, both scales are in arbitrary units (a.u.).

The blue line in Figure 1C represents the 45° diagonal line, where the original and bias-corrected image intensities are equal ($X = Y$), and the pink line represents a generalized additive model (GAM) (Hastie and Tibshirani, 1990) scatterplot smoother estimate obtained using the **mgcv** package (Wood, 2011). We see that for values in the low range of the data (< 10), the T1 values and bias-corrected T1 values, on average, fall along the diagonal, but in the higher range the bias-corrected values are lower.

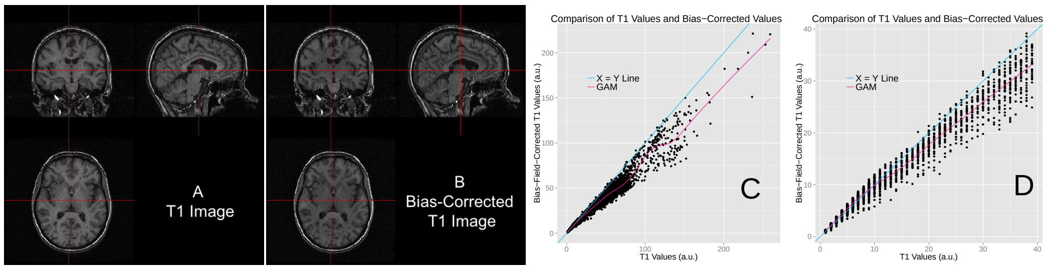

A
T1 Image
B
Bias-Corrected
T1 Image
Comparison of T1 Values and Bias−Corrected Values
X = Y Line
GAM
Bias−Field−Corrected T1 Values (a.u.)
T1 Values (a.u.)
C
D

Figure 1: Results of inhomogeneity correction. We present the original T1 image (A), bias-corrected T1 image (B), and the scatterplot of the sampled values comparing the values from the T1 image to the bias-corrected values (C). We see in Panel C for values in the low range of the data (< 10), the T1 values and bias-corrected T1 values, on average, fall along the diagonal (blue line), which is further illustrated in Panel D, which plots values < 40. Values > 10 for the original T1 image are lower than the bias-corrected T1 values shown by a generalized additive model (GAM) smoother (pink line, Panel C).

Within-visit co-registration

All subsequent steps will be performed on the bias-corrected images. We will first co-register the images within each separate visit to the T1 image from that visit. This operation overlays the images on one another and allows us to investigate joint distributions of voxel intensities from different image modalities. This is performed using FMRIB's Linear Image Registration Tool (FLIRT; Jenkinson and Smith 2001; Jenkinson et al. 2002). As the images are from the same individual, we may assume that the overall shape of the brain has not changed, but each scan may have undergone a translation and/or rotation in space. Therefore, we will use a rigid-body transformation, with 6 degrees of freedom (dof).

The **fslr** command `flirt` calls the FSL command `flirt`, taking the input image (`infile`) and the reference image that serves as a template (`reffile`). Any additional options for FLIRT can be passed using the `opts` argument. We will use the defaults (i.e. trilinear interpolation) and the `-v` option for diagnostic messages to be printed. Since we are doing a rigid-body transformation, we set the degrees of freedom (`dof`) to 6. Here we present the code for registering the baseline T2 image to the baseline T1 image; we will subsequently repeat this process for the baseline FLAIR and PD images and for the follow-up scans.

```
flirt(reffile = "01-Baseline_T1_FSL_BiasCorrect",
      infile = "01-Baseline_T2_FSL_BiasCorrect",
      omat = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1.mat",
      dof = 6,
      outfile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1",
      opts = "-v")
```

The resulting image transformation is stored using the file name passed to the `omat` (**o**utput **mat**rix) argument. This matrix can be used to transform other images, that were in the same space as the input image, to the reference image space. The **fslr** package will currently only return '`nifti`' objects, and not a list of objects, such as the output image, transformation matrix, etc. Thus, any transformation files that are needed after the command is executed must be specified.

After co-registration, one could compare images of different modalities at the same voxels, such as T1 versus FLAIR images, which is presented in Figure 2. The images are presented at the same cross section for the baseline T1 (Panel 2A) and FLAIR (Panel 2B) images. The same brain areas are presented in each modality, indicating adequate registration.

In the previous example, we presented a rigid-body transformation, using the default parameters. `flirt` has options for different cost functions to optimize over, interpolation operators to estimate voxel intensity, and additional degrees of freedom for performing affine transformations. These options can be passed to the FSL `flirt` command using the `opts` argument in the **fslr** `flirt` function.

Note that each **fslr** function has a corresponding help function, which is the **fslr** command appended with `.help()`, which prints out the FSL help page for that function. For example, users can see which options can be changed in the FSL `flirt` command by executing the `flirt.help()` function. Additional non-linear registration techniques are presented in Section "Registration to the MNI template".

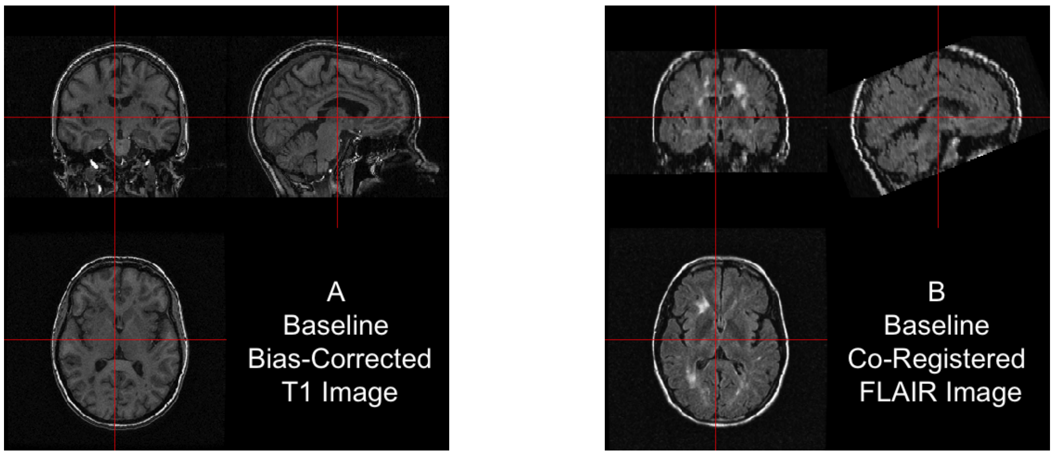


Figure 2: Results of within-visit co-registration. We present the bias-corrected T1 image (A) and the co-registered bias-corrected FLAIR image (B).

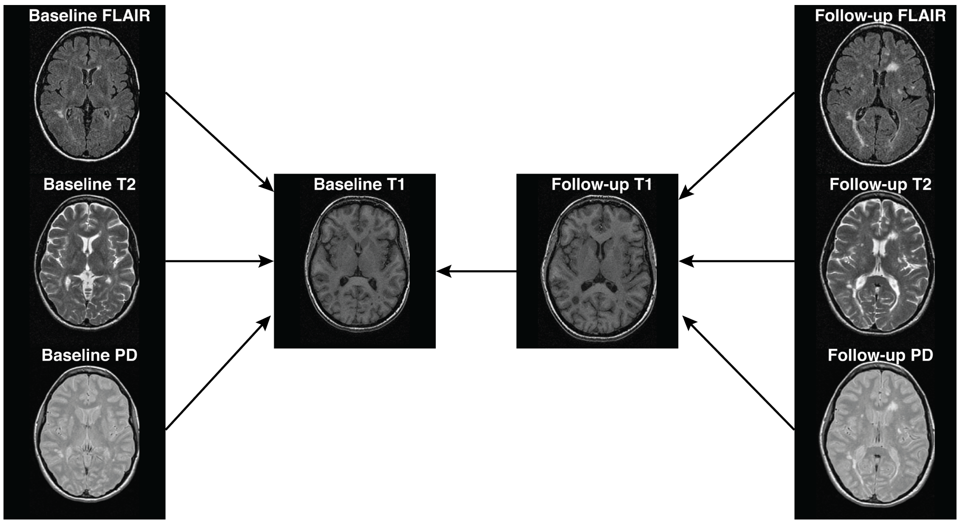


Figure 3: Between-visit registration process. First, we registered all scans within a visit (baseline or follow-up) to the T1 image. We then registered the follow-up T1 image to the baseline T1 image and applied the transformation to the follow-up T2, FLAIR, and PD images previously co-registered to the follow-up T1 image.

Between-visit co-registration

Though across-modality comparisons can be achieved by performing within-visit co-registration, across-visit registration is required for assessing within-modality differences between longitudinal scans. To compute difference images, we co-register follow-up images to the baseline images within each modality. Similar to the within-visit co-registration, we use a rigid-body transformation. We will register the T1 images from baseline and follow-up, and apply this transformation to the co-registered-to-T1 images from above (see Figure 3 for illustration).

Though this registration involves two interpolations of the data and may not be optimal for within-modality comparisons, we have already obtained the co-registered-to-T1 images in Section "Within-visit co-registration" and must perform only one additional registration. This operation also demonstrates how to apply transformation matrices in **fslr**. Here we register the follow-up T1 image to the baseline T1 image, again using a rigid-body transformation (6 dof):

```
flirt(reffile = "01-Baseline_T1_FSL_BiasCorrect",
      infile = "01-Followup_T1_FSL_BiasCorrect",
      omat = "01-Followup_T1_FSL_BiasCorrect_rigid_to_BaseT1.mat",
      dof = 6,
      outfile = "01-Followup_T1_FSL_BiasCorrect_rigid_to_BaseT1",
      opts = "-v")
```

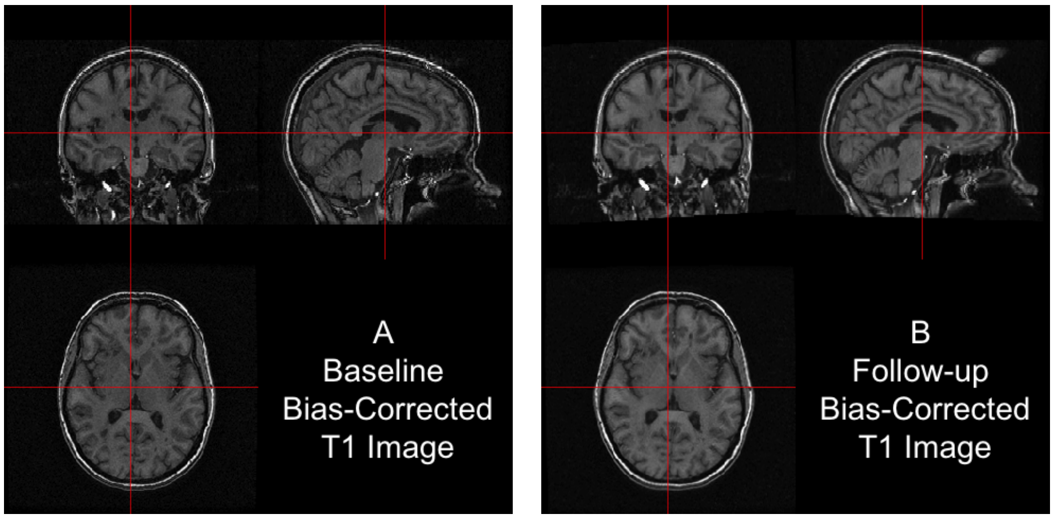


Figure 4: Results from FLIRT. The bias-corrected baseline T1 is presented in (A) and the registered bias-corrected follow-up T1 is presented in (B), each displayed at the same intersection. We observe that the observed images correspond to the same brain area, indicating a good registration.

Now, both T1 images are aligned in the space of the baseline T1 image. We present the results in Figure 4: the bias-corrected baseline T1 image in Panel A and the co-registered bias-corrected follow-up T1 in Panel B. The images displayed at the same cross section correspond to the same brain area, indicating a good registration.

Using the `flirt_apply` function from **fslr**, we can apply the transformation matrix to the T2, PD, and FLAIR images from the follow-up visit, previously co-registered to the T1 from follow-up, to align them to the baseline T1 image space. The code below aligns the follow-up T2 image, previously registered to the follow-up T1 image, to the baseline T1 image:

```
flirt_apply(reffile = "01-Baseline_T1_FSL_BiasCorrect", # register to this
            infile = "01-Followup_T2_FSL_BiasCorrect_rigid_to_T1", # reg to Followup T1
            initmat = "01-Followup_T1_FSL_BiasCorrect_rigid_to_BaseT1.mat", #transform
            outfile = "01-Followup_T2_FSL_BiasCorrect_rigid_to_BaseT1" # output file
            )
```

In Figure 5, we display each image after FLIRT has been applied. Each image is in the baseline T1 image space, displayed at the same cross section. Each panel shows the same brain areas across modalities, indicating adequate registration. We see that some areas of the brain are cropped from the field of view, which may be problematic if relevant brain areas are removed. We have registered all images with the skull and extracranial tissue included. A better method may be to perform registration on brain tissues only, in which case we must perform brain extraction.

Brain extraction

The process of extracting brain tissue from the acquired image, referred to as brain extraction or skull stripping, is a crucial step in many analyses. We will perform brain extraction using FSL's brain extraction tool (BET; Smith 2002; Jenkinson et al. 2005) using parameters recommended by Popescu et al. (2012), which were derived from patients with MS. No other published R package on CRAN (e.g. **AnalyzeFMRI**, **RNiftyReg**, or **fmri**) has brain extraction functionality for brain imaging. Other neuroimaging software provide brain extraction, such as AFNI (Cox, 1996), SPM (Ashburner and Friston, 2005), and Freesurfer (Fischl, 2012), and multi-atlas label fusion techniques (Doshi et al., 2013).

```
fslbet(infile =  "01-Baseline_T1",
       outfile = "01-Baseline_T1_FSL_BiasCorrect_Brain",
       opts = "-B -f 0.1 -v",  # from Popescu et al.
       betcmd = "bet",
       intern = FALSE)
```

We ran BET on the non-corrected T1 image as the `-B` option performs inhomogeneity correction from FAST as part of the procedure. The option `-f 0.1` denotes the fractional intensity (FI) parameter

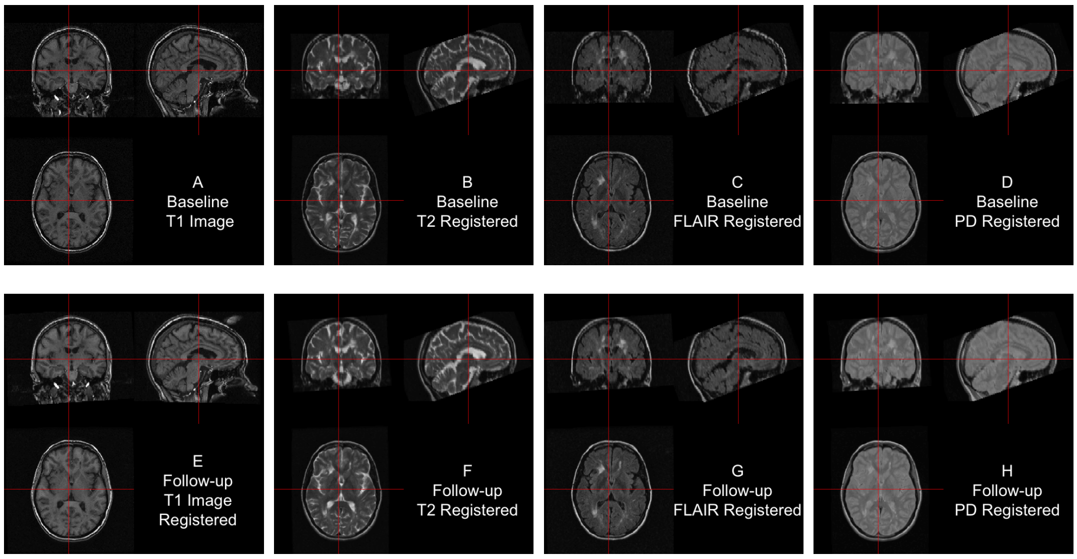


Figure 5: Between-visit registration results. The complete set of acquired images, first co-registered within visit to the T1 image of that visit, then registered to the baseline T1 image using the follow-up T1 to baseline T1 transformation matrix. All registrations performed rigid-body transformations.

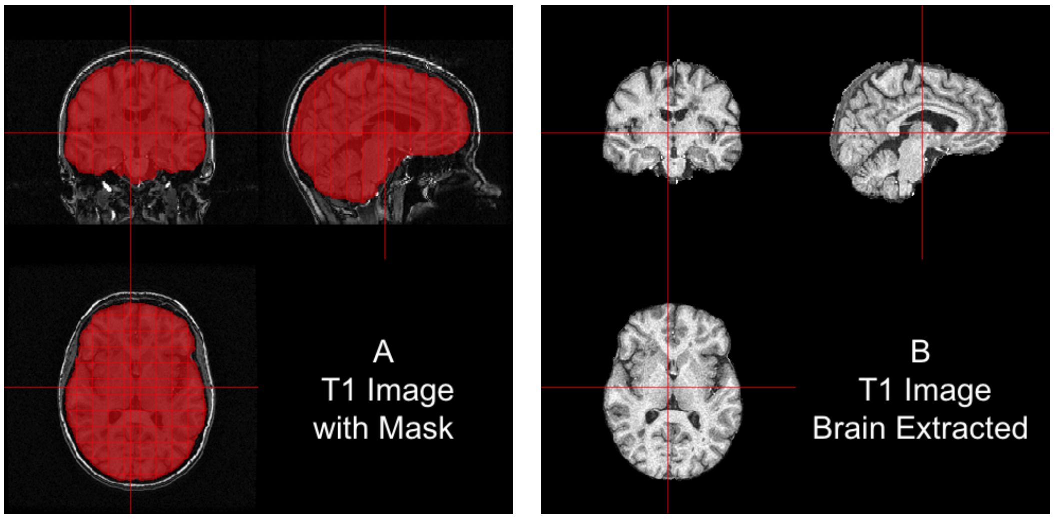


Figure 6: Results from BET. In (A), we show the bias-corrected T1 image with the mask from BET overlaid in red. In (B), we display the extracted brain. We see that the brain extraction performed well, not including any areas of the skull or the neck while not discarding large areas of the brain.

in BET: it varies between 0 and 1 and determines the location of the edge of the segmented brain image; smaller values correspond to larger brain masks. In Figure 6, the bias-corrected T1 image is shown with the brain mask overlaid in red (Panel A) and the resulting masked brain (Panel B). We see that the brain extraction performed well, not including any areas of the skull or the neck while not discarding brain tissue. Towards the back of the brain, some areas of the subarachnoid space remain, which may be unacceptable for certain analyses, such as estimation of the volume of brain tissue.

Note that `fslbet` writes both a file containing the brain-extracted image and another image containing the binary brain mask. As all other images are registered to the baseline T1 space, we can use this mask to extract the brain from other images, such as the baseline T2 image, using the **fslr** function `fslmask`. In this example, we mask the registered-to-T1, bias-corrected T2 image with the binary brain mask, save the image to the filename specified in `outfile`, and also set the `retimg` option to `TRUE`, indicating the `fslmask` command to **ret**urn the **im**age. The returned object is a '`nifti`' object, assigned to the R object `mask`:

```
mask <- fslmask(file = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1",
                mask = "01-Baseline_T1_FSL_BiasCorrect_Brain_mask",
                outfile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain",
                retimg = TRUE)
```

We now have all images in the same stereotaxic space with the brain extracted.

Registration to the MNI template

In many studies, information is aggregated across a population of images from different participants. For the information to have the same interpretation spatially across participants, images from all participants need to be aligned in the same stereotaxic space ("template" space), requiring registration to a standard template image. A frequently used set of templates are provided by MNI (Montreal Neurological Institute). We have registered the baseline T1 image to the MNI T1 template (Grabner et al., 2006), included with FSL. As an individual's brain does not necessarily have the same size as the template, it is not appropriate to use rigid-body transformations. Instead, non-linear transformations are needed. As these are patients with MS and have lesions, different non-linear registrations to template space can have a large impact on the outcome of analysis (see Eloyan et al. 2014 for discussion).

We will first register the baseline T1 image to the T1 template using an affine registration, which can perform scaling and shearing operations in addition to translation and rotation. Although an affine transformation has more degrees of freedom than a rigid transformation, it may not provide a registration sufficient for analysis. We will then use FNIRT (FMRIB's Nonlinear Image Registration Tool) to achieve better overlap of local brain structures (Jenkinson et al., 2012; Andersson et al., 2007). As we are concerned with good overlap only in brain structures, and not in areas such as the skull, we will register the brain-extracted brain images to the brain-only template. The **fslr** function `fnirt_with_affine` will register using `flirt` with an affine transformation and then non-linearly register this image to the template using `fnirt`. If this affine transformation is not executed before `fnirt`, the image will not achieve the necessary scaling into the template space.

```
fnirt_with_affine(infile = "01-Baseline_T1_FSL_BiasCorrect_Brain",
                  reffile = file.path(fsldir(), "data", "standard",
                                      "MNI152_T1_1mm_brain"),
                  flirt.omat = "01-Baseline_T1_FSL_BiasCorrect_Brain_affine_toMNI.mat",
                  flirt.outfile = "01-Baseline_T1_FSL_BiasCorrect_Brain_affine_toMNI",
                  outfile = "01-Baseline_T1_FSL_BiasCorrect_Brain_toMNI")
```

The results of the registration can be seen in Figure 7. Each panel represents a different axial slice ($z = 25$, 45, 92, or 137) in the template space of the template image (A, C, E, G) or the registered T1 image (B, D, F, H). Each slice shows the registered T1 image has similar brain structure represented in the same area as the template image, indicating good registration.

Applying transformations to co-registered data

Since all the data is represented in the same image space, we can apply the estimated affine transformation and non-linear warping coefficient field to each image to represent that image in template space. The affine transformation must be applied with `flirt_apply` and the warping coefficient using `fsl_applywarp`, which calls the FSL `applywarp` function.

Here we present the application of the transformations to the baseline T2 image, previously registered to the baseline T1.

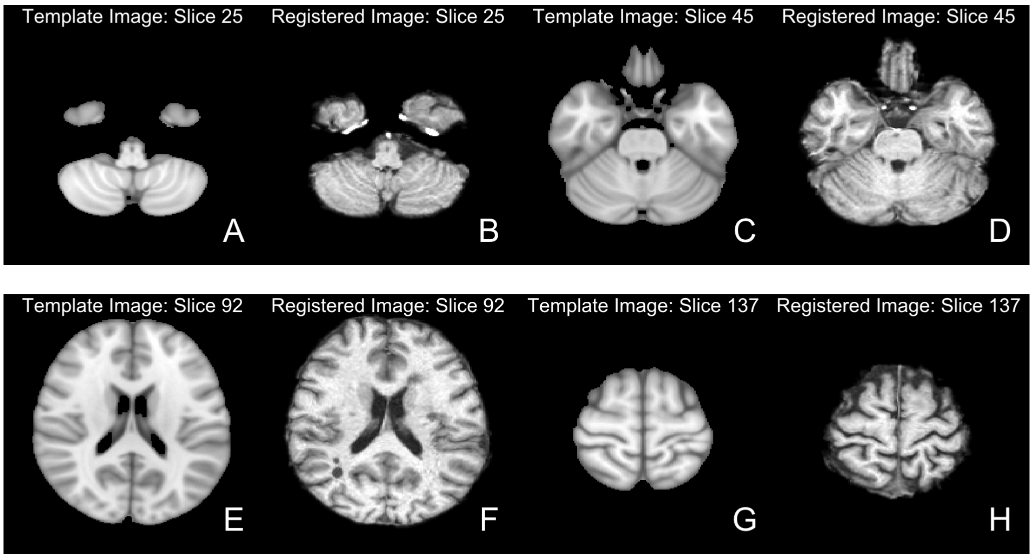

Template Image: Slice 25
Registered Image: Slice 25
Template Image: Slice 45
Registered Image: Slice 45
A
B
C
D
Template Image: Slice 92
Registered Image: Slice 92
Template Image: Slice 137
Registered Image: Slice 137
E
F
G
H

Figure 7: Results from FNIRT. We present different axial slices of the template (A, C, E, G) and the registered T1 image (B, D, F, H). The slices represented are 25 (A, B), 45 (C, D), 92 (E, F) and 137 (G, H). We note that areas of the brain coincide between the template and registered image.

```
flirt_apply(infile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain",
            reffile = file.path(fsldir(), "data", "standard", "MNI152_T1_1mm_brain"),
            initmat = "01-Baseline_T1_FSL_BiasCorrect_Brain_affine_toMNI.mat",
            outfile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain_toMNI")
fsl_applywarp(infile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain_toMNI",
              reffile = file.path(fsldir(), "data", "standard", "MNI152_T1_1mm_brain"),
              warpfile = "01-Baseline_T1_FSL_BiasCorrect_Brain_affine_toMNI_warpcoef",
              outfile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain_toMNI")
```

These two operations can also be performed in a single call to the **fslr** `fnirt_with_affine_apply` function.

With multiple participants, this process yields a multi-person, multi-modal, longitudinal imaging dataset that can be used for analyses.

Conclusion

The neuroimaging community has developed a large collection of tools for image processing and analysis. R has a number of packages to perform operations on images; **EBImage** is one good example (Pau et al., 2010). Much of the fundamental functionality of neuroimage processing is not currently available in R, such as brain extraction and tissue segmentation. We present **fslr** to provide R users functions for image processing and analysis that are based on FSL, an established image processing and analysis software suite. Interfacing R with existing, powerful software provides users with thoroughly-tested software and an additional community of users, which would not be available if the functions were rewritten in R. **fslr** should be easy to use for any standard R user; the workflow allows R users to manipulate array-like '`nifti`' objects, pass them to **fslr** functions, which return '`nifti`' objects. Moreover, as FSL and R are open-source and free, this software is readily available to all users.

There has been an increasing popularity of similar interfacing of tools within the Python community such as Nipype (Gorgolewski et al. 2011; `https://qa.debian.org/popcon.php?package=nipype`). As many users of R may not have experience with Python or bash scripting, we believe **fslr** provides a lower threshold for use in the R community. Other packages provide R users additional neuroimaging processing functionality such as **AnalyzeFMRI**, **RNiftyReg**, and **fmri**.

For example, other inhomogeneity correction methods exist, such as the popular N3 (Sled et al., 1998) and N4 (Tustison et al., 2010), methods which are not implemented in **fslr**. **ANTsR** (Avants et al., 2015) is an R package that interfaces with the ANTs (advanced normalization tools) software suite (Avants et al., 2011). ANTs has implementations of these correction methods, an increased set of registration techniques, and other methods for image processing. Other packages such as this, along with **fslr**, can create a diverse set of tools for neuroimaging within R, building on preexisting and widely-accepted software.

Most importantly, as **fslr** is based on the R framework, all the benefits of using R are available, such as dynamic documents, reproducible reports, customized figures, and state-of-the-art statistical methods. These benefits provide unique functionality compared to other software packages for neuroimaging.

All data and code processed here is located at https://github.com/muschellij2/FSLR_Data.

Supplemental material

Smoothing images

Let us show how to pass a Gaussian smoother over an image using `fslsmooth` (FSL `fslmaths -s` function). First we will read in the registered-to-template baseline T1 brain image:

```
t1_to_temp <- readNIfTI("01-Baseline_T1_FSL_BiasCorrect_Brain_toMNI", reorient = FALSE)
```

We will smooth the image using a Gaussian smoother with $\sigma = 3\text{mm}^3$:

```
smooth <- fslsmooth(t1_to_temp, sigma = 3, retimg = TRUE)
```

The result is presented in Figure 8A.

Thresholding images

The **fslr** `fslbin` function will binarize a '`nifti`' image object: all values ≤ 0 are set to 0, and set to 1 otherwise. Let us binarize the registered image:

```
binned <- fslbin(t1_to_temp, retimg = TRUE)
```

The result is presented in Figure 8B.

The **fslr** `fslthresh` function provides more control for thresholding by setting a lower threshold (`thresh` argument) and upper threshold (`uthresh` argument). These thresholds are inclusive, and will set values less than (but not equal to) `thresh` or greater than (but not equal to) `uthresh` to 0. Voxels with values between `thresh` and `uthresh` (inclusive) will be returned as their original value. Let us threshold the smoothed image between 30 and 50:

```
thresh <- fslthresh(t1_to_temp, thresh = 30, uthresh = 50, retimg = TRUE)
```

The result is presented in Figure 8C.

Eroding and dilating images

In many applications, one wants to erode (i.e. shrink) an image mask. The **fslr** function `fslerode` performs this operation. Note, if no options are specified for the kernel (in the `kopts` argument), the default $3 \times 3 \times 3$ voxel box kernel is used. Here we erode the binarized image from above and plot the voxels eroded:

```
eroded <- fslerode(binned, retimg = TRUE)
```

The result is presented in Figure 8D. If one inverts the binary mask, performs erosion, and then inverts the resulting erosion mask, this procedure is equivalent to dilation. The **fslr** function `fsldilate` (version 1.4 and above) will also perform image dilation.

Extracting image information

Although most **fslr** functions provide an image as the designated output, one may wish to extract image information from a NIfTI file on disk, without reading it into R. The **fslr** `fslstats`, `fslval`, and `fslhd` functions are particularly useful and call functions of the same name from FSL.

For example, we can extract the number of slices in the third dimension of the bias-corrected T1 image using `fslval`:

```
fslval("01-Baseline_T1_FSL_BiasCorrect_Brain", keyword = "dim3")
```

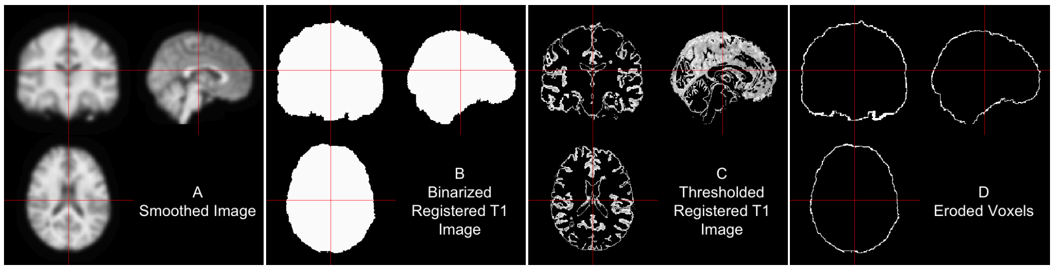


Figure 8: Results of fslr functions. We present the smoothed registered-to-template T1 image (A), binarized image (B), thresholded image (C), and image of the eroded voxels after eroding the binarized image (D).

```
[1] "124"
```

We could also extract the entire header using `fslhd` and assign it to an object:

```
img_hdr <- fslhd("01-Baseline_T1_FSL_BiasCorrect_Brain")
```

We can extract the mean of the image of non-zero voxels:

```
fslstats("01-Baseline_T1_FSL_BiasCorrect_Brain", opts = "-M")
```

```
[1] "51.264791"
```

Overall, there are many functions and options that allow the user to compute statistics or obtain header information from an image on disk without having to load it into R, which can reduce computation time.

Additional fslr functionality

Although the main goal of **fslr** is to interface R and FSL, there is a set of functions in **fslr** that are not designed to interface with FSL, but rather provide helper functions for '`nifti`' objects from the **oro.nifti** package. We will display 2 example functions: `cal_img` and `niftiarr`. The `cal_img` function resets the `cal_min` and `cal_max` slots on a '`nifti`' object, which are used to determine colors when plotting. The `niftiarr` function copies a '`nifti`' object and replaces the `.Data` slot, which contains the actual image intensity values, with a provided array.

Let us illustrate by discussing 2 ways to mask an image. We will read in the bias-corrected T1 image and the mask from BET:

```
base_t1 <- readNIfTI("01-Baseline_T1_FSL_BiasCorrect", reorient = FALSE)
base_t1_mask <- readNIfTI("01-Baseline_T1_FSL_BiasCorrect_Brain_mask",
  reorient = FALSE)
```

One way to mask the T1 image is to multiply the image by the binary mask:

```
base_t1_1 <- base_t1 * base_t1_mask
class(base_t1_1)
```

```
[1] "array"
```

We see that the resulting object is an array and not a '`nifti`' object (as of **oro.nifti** version `0.4.3`). This may be a problem when trying to plot or manipulate this object using methods for '`nifti`' objects. To address this problem, the `niftiarr` function in **fslr** inputs a '`nifti`' object and an `array`, and returns a '`nifti`' object with the provided `array` in the `.Data` slot, copying over the image information from the input '`nifti`' object. As of **oro.nifti** version `0.5.0`, the output from above will be a '`nifti`' object, but the function explained below, `niftiarr`, is still of use in cases when creating a new '`nifti`' object.

```
base_t1_1 <- niftiarr(base_t1, base_t1_1)
class(base_t1_1)
```

```
[1] "nifti"
attr(,"package")
[1] "oro.nifti"
```

Another way of masking the image is to subset the values of the image that are not in the mask and setting those values to 0 (or some other value).

```
base_t1_2 <- base_t1
base_t1_2[base_t1_mask == 0] <- 0
class(base_t1_2)
```

```
[1] "nifti"
attr(,"package")
[1] "oro.nifti"
```

We see that this correctly returns an object of class 'nifti'. One problem is that the we have changed the data in the 'nifti' object base_t1_2 but did not reset the other slots in this object to reflect this change.

In a 'nifti' object, the cal_min and cal_max slots equal the minimum and maximum values, respectively, of the data. The orthographic function (from **oro.nifti**) uses these values for plotting; also, if these slots do not equal the minimum and maximum, the writeNIfTI function (from **oro.nifti**) will fail. The cal_img is a simple helper function that will set the cal_min and cal_max slots to the correct values. Let us look at the range of the data and the cal_min and cal_max slots:

```
range(base_t1_2)
```

```
[1]   0.0000 409.3908
```

```
c(base_t1_2@cal_min, base_t1_2@cal_max)
```

```
[1] 0 0
```

An issue with the readNIfTI function from **oro.nifti** is that the cal_min and cal_max slots may be both read as zero. Let us set these to the range using the cal_img command from **fslr**:

```
base_t1_2 <- cal_img(base_t1_2)
c(base_t1_2@cal_min, base_t1_2@cal_max)
```

```
[1]   0.0000 409.3908
```

We see that after these operations are done in different ways, the resulting 'nifti' objects are equivalent.

```
identical(base_t1_1, base_t1_2)
```

```
[1] TRUE
```

Additional helper functions such as these are included in **fslr** for plotting and image manipulation.

Bibliography

J. L. Andersson, M. Jenkinson, and S. Smith. Non-linear registration, aka Spatial normalisation. FMRIB technical report TR07JA2, FMRIB Analysis Group of the University of Oxford, 2007. URL http://fmrib.medsci.ox.ac.uk/analysis/techrep/tr07ja2/tr07ja2.pdf. [p169]

J. Ashburner and K. J. Friston. Unified segmentation. *NeuroImage*, 26(3):839–851, 2005. [p167]

B. B. Avants, N. J. Tustison, G. Song, P. A. Cook, A. Klein, and J. C. Gee. A reproducible evaluation of ANTs similarity metric performance in brain image registration. *NeuroImage*, 54(3):2033–2044, Feb. 2011. [p170]

B. B. Avants, B. M. Kandel, J. T. Duda, P. A. Cook, and N. J. Tustison. *ANTsR: An R Package Providing ANTs Features in R*, 2015. URL http://stnava.github.io/ANTsR/. R package version 0.3.1. [p170]

C. Bordier, M. Dojat, and P. L. de Micheaux. Temporal and spatial independent component analysis for fMRI data sets embedded in the AnalyzeFMRI R package. *Journal of Statistical Software*, 44(9): 1–24, 2011. URL http://www.jstatsoft.org/v44/i09/. [p163]

J. Carp. The secret lives of experiments: Methods reporting in the fMRI literature. *NeuroImage*, 63(1): 289–300, Oct. 2012. [p163]

J. Clayden. *RNiftyReg: Medical Image Registration Using the NiftyReg Library*, 2015. URL http://CRAN.R-project.org/package=RNiftyReg. R package version 1.1.3, based on original code by Marc Modat and Pankaj Daga. [p163]

R. W. Cox. AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical Research*, 29(3):162–173, 1996. [p167]

J. Doshi, G. Erus, Y. Ou, B. Gaonkar, and C. Davatzikos. Multi-atlas skull-stripping. *Academic Radiology*, 20(12):1566–1576, Dec. 2013. [p167]

A. Eloyan, H. Shou, R. T. Shinohara, E. M. Sweeney, M. B. Nebel, J. L. Cuzzocreo, P. A. Calabresi, D. S. Reich, M. A. Lindquist, and C. M. Crainiceanu. Health effects of lesion localization in multiple sclerosis: Spatial registration and confounding adjustment. *PLoS ONE*, 9(9):e107263, Sept. 2014. [p169]

B. Fischl. FreeSurfer. *NeuroImage*, 62(2):774–781, 2012. [p167]

K. Gorgolewski, C. D. Burns, C. Madison, D. Clark, Y. O. Halchenko, M. L. Waskom, and S. S. Ghosh. Nipype: A flexible, lightweight and extensible neuroimaging data processing framework in Python. *Frontiers in Neuroinformatics*, 5(13), 2011. [p170]

G. Grabner, A. L. Janke, M. M. Budge, D. Smith, J. Pruessner, and D. L. Collins. Symmetric atlasing and model based segmentation: An application to the hippocampus in older adults. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. P. Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, R. Larsen, M. Nielsen, and J. Sporring, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006*, volume 4191, pages 58–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. [p169]

R. Guillemaud and M. Brady. Estimating the bias field of MR images. *IEEE Transactions on Medical Imaging*, 16(3):238–251, 1997. [p164]

T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*, volume 43. CRC Press, 1990. [p164]

M. Jenkinson and S. Smith. A global optimisation method for robust affine registration of brain images. *Medical Image Analysis*, 5(2):143–156, 2001. [p163, 165]

M. Jenkinson, P. Bannister, M. Brady, and S. Smith. Improved optimization for the robust and accurate linear registration and motion correction of brain images. *NeuroImage*, 17(2):825–841, 2002. [p163, 165]

M. Jenkinson, M. Pechaud, and S. Smith. BET2: MR-based estimation of brain, skull and scalp surfaces. In *Eleventh Annual Meeting of the Organization for Human Brain Mapping*, volume 17, 2005. URL `http://mickaelpechaud.free.fr/these/HBM05.pdf`. [p167]

M. Jenkinson, C. F. Beckmann, T. E. J. Behrens, M. W. Woolrich, and S. M. Smith. FSL. *NeuroImage*, 62 (2):782–790, Aug. 2012. [p163, 169]

G. Pau, F. Fuchs, O. Sklyar, M. Boutros, and W. Huber. EBImage—an R package for image processing with applications to cellular phenotypes. *Bioinformatics*, 26(7):979–981, 2010. [p170]

V. Popescu, M. Battaglini, W. S. Hoogstrate, S. C. J. Verfaillie, I. C. Sluimer, R. A. van Schijndel, B. W. van Dijk, K. S. Cover, D. L. Knol, M. Jenkinson, F. Barkhof, N. de Stefano, and H. Vrenken. Optimizing parameter choice for FSL-Brain Extraction Tool (BET) on 3d T1 images in multiple sclerosis. *NeuroImage*, 61(4):1484–1494, July 2012. [p167]

J. G. Sled, A. P. Zijdenbos, and A. C. Evans. A nonparametric method for automatic correction of intensity nonuniformity in MRI data. *IEEE Transactions on Medical Imaging*, 17(1):87–97, 1998. [p170]

S. M. Smith. Fast robust automated brain extraction. *Human Brain Mapping*, 17(3):143–155, Nov. 2002. [p163, 167]

E. M. Sweeney, R. T. Shinohara, C. D. Shea, D. S. Reich, and C. M. Crainiceanu. Automatic lesion incidence estimation and detection in multiple sclerosis using multisequence longitudinal MRI. *American Journal of Neuroradiology*, 34(1):68–73, Jan. 2013. [p164]

K. Tabelow and J. Polzehl. Statistical parametric maps for functional MRI experiments in R: The package fmri. *Journal of Statistical Software*, 44(11):1–21, 2011. URL `http://www.jstatsoft.org/v44/i11/`. [p163]

N. J. Tustison, B. B. Avants, P. A. Cook, Y. Zheng, A. Egan, P. A. Yushkevich, and J. C. Gee. N4itk: Improved N3 bias correction. *IEEE Transactions on Medical Imaging*, 29(6):1310–1320, 2010. [p170]

B. Whitcher, V. J. Schmid, and A. Thornton. Working with the DICOM and NIfTI data standards in R. *Journal of Statistical Software*, 44(6):1–28, 2011. URL `http://www.jstatsoft.org/v44/i06/`. [p163]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2009. URL http://had.co.nz/ggplot2/book. [p164]

S. N. Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(1):3–36, 2011. [p164]

Y. Zhang, M. Brady, and S. Smith. Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. *IEEE Transactions on Medical Imaging*, 20 (1):45–57, 2001. [p163, 164]

John Muschelli
PhD Student
Johns Hopkins Bloomberg School of Public Health,
Baltimore, MD 21231
USA
jmuschel@jhsph.edu

Elizabeth Sweeney
PhD Student
Johns Hopkins Bloomberg School of Public Health,
Baltimore, MD 21231
USA
Special Volunteer Translational Neuroradiology Unit, Neuroimmunology Branch, National Institute of Neurological Disease and Stroke, National Institute of Health,
Bethesda, MD 20892
USA
emsweene@jhsph.edu

Martin Lindquist
Associate Professor
Johns Hopkins Bloomberg School of Public Health,
Baltimore, MD 21231
USA
mlindqui@jhsph.edu

Ciprian Crainiceanu
Professor
Johns Hopkins Bloomberg School of Public Health,
Baltimore, MD 21231
USA
ccrainic@jhsph.edu

Identifying Complex Causal Dependencies in Configurational Data with Coincidence Analysis

by Michael Baumgartner and Alrik Thiem

Abstract We present **cna**, a package for performing Coincidence Analysis (CNA). CNA is a configurational comparative method for the identification of complex causal dependencies—in particular, causal chains and common cause structures—in configurational data. After a brief introduction to the method's theoretical background and main algorithmic ideas, we demonstrate the use of the package by means of an artificial and a real-life data set. Moreover, we outline planned enhancements of the package that will further increase its applicability.

Introduction

Configurational comparative methods (CCMs) subsume techniques for the identification of complex causal dependencies in configurational data using the theoretical framework of Boolean algebra and its various extensions (Rihoux and Ragin, 2009). For example, Qualitative Comparative Analysis (QCA; Ragin, 1987, 2000, 2008)—hitherto the most prominent representative of CCMs—has been applied in areas as diverse as business administration (e.g., Chung, 2001), environmental science (van Vliet et al., 2013), evaluation (Cragun et al., 2014), political science (Thiem, 2011), public health (Longest and Thoits, 2012) and sociology (Crowley, 2013). Besides three stand-alone programs based on graphical user interfaces, three R packages for QCA are currently available, each with a different scope of functionality: **QCA** (Dusa and Thiem, 2014; Thiem and Duşa, 2013a,c), **QCA3** (Huang, 2014) and **SetMethods** (Quaranta, 2013; an add-on package to Schneider and Wagemann, 2012).

A novel technique called Coincidence Analysis (CNA) has recently joined the family of CCMs (Baumgartner, 2009a,b, 2013a). Like QCA, CNA searches for rigorously minimized sufficient and necessary conditions of causally modeled outcomes, and it implements the same regularity theory of causation as QCA, that is, the theory most prominently advanced by Mackie (1974). Contrary to QCA, however, CNA can treat any number of factors in a processed data set as endogenous (outcomes), and it does not eliminate redundancies from sufficient and necessary conditions by means of Quine-McCluskey optimization (Quine, 1959; McCluskey, 1965), but by means of an optimization algorithm that is custom-built for causal modeling. As a direct consequence of these differences, CNA can identify common cause and causal chain structures. Moreover, the algorithm does not need to be told which factors are endogenous and which ones exogenous; it can infer that from the data. What is more, limited data diversity does not force CNA to resort to counterfactual additions to the data. Finally, while the QCA programs *fs/QCA* (Ragin and Davey, 2014), *fuzzy* (Longest and Vaisey, 2008), *Tosmana* (Cronqvist, 2011) and *Kirq* (Reichert and Rubinson, 2014) often fail to find all data-fitting models (cf. Baumgartner and Thiem, 2014; Thiem, 2014c; Thiem and Duşa, 2013b), the R implementation of CNA presented in this paper (**cna**; Ambuehl et al., 2015) not only ensures—as does **QCA**—that all single-outcome models are identified but additionally recovers the whole space of multiple-outcome models that fit the data.

After an introduction to the theoretical and algorithmic background of CNA, we demonstrate the potential of the **cna** R package by means of an artificial and a real-life data set. In the final section, we outline planned enhancements of **cna** that will further increase its applicability.

Theoretical and algorithmic background

CCMs search for causal dependencies as defined by so-called *regularity theories* of causation, whose development dates back to David Hume (1711-1776) and John Stuart Mill (1806-1873). By implementing techniques of Boolean algebra, modern regularity theories spell out the notion of causal relevance in terms of redundancy-free (minimized) sufficiency and necessity relations among the elements of analyzed sets of factors (Mackie, 1974; Graßhoff and May, 2001; Baumgartner, 2008, 2013b).

The crucial component of the regularity theoretic definiens of causal relevance is the notion of a minimal theory. A *minimal theory* of a factor Z is a minimally necessary disjunction of minimally sufficient conditions of Z. A conjunction Φ of coincidently instantiated factors, i.e. $Y_1 * Y_2 * \ldots * Y_n$, is a minimally sufficient condition of Z if, and only if (iff), Φ is sufficient for Z ($\Phi \rightarrow Z$), and there exists no proper part Φ' of Φ such that $\Phi' \rightarrow Z$. A proper part Φ' of Φ is the result of eliminating at least one conjunct from Φ. A disjunction Ψ of minimally sufficient conditions, i.e. $\Phi_1 + \Phi_2 + \ldots + \Phi_n$, is a minimally necessary condition of Z iff Ψ is necessary for Z ($Z \rightarrow \Psi$), and there exists no proper part Ψ' of Ψ such that $Z \rightarrow \Psi'$. A proper part Ψ' of Ψ is the result of eliminating at least one disjunct from Ψ. Overall, a minimal theory of Z has the following biconditional form: $\Psi \leftrightarrow Z$ (where Ψ is an expression in disjunctive normal form and Z is a single factor).

A minimal theory represents the causally interpretable dependencies of sufficiency and necessity among the factors contained in a data set δ. That is, causal relevance can be defined in terms of membership in a minimal theory, or more specifically: a factor A is causally relevant to a factor B in a data set δ iff δ entails a minimal theory $\Psi \leftrightarrow B$ such that A is contained in Ψ.

CNA aims to infer minimal theories from δ by, first, identifying sufficient and necessary conditions in δ and by, second, minimizing those conditions. To the latter end, CNA tests the redundancy of factors by eliminating them from sufficient and necessary conditions and checking whether the remaining conditions are still sufficient and necessary, respectively. More specifically, to determine whether a sufficient condition $Y_1{*}Y_2{*}\ldots{*}Y_h$ of a factor Z is minimally sufficient, CNA systematically eliminates conjuncts from $Y_1{*}Y_2{*}\ldots{*}Y_h$. For each conjunction that results from such an elimination, say for $Y_2{*}Y_3{*}\ldots{*}Y_h$, CNA then parses the processed data δ to check whether δ contains $Y_2{*}Y_3{*}\ldots{*}Y_h$ in combination with the negation of Z, i.e. z. If δ does not contain such a configuration, $Y_2{*}Y_3{*}\ldots{*}Y_h$ is itself sufficient for Z, which means that Y_1 is redundant. CNA then proceeds to eliminate the next conjunct from $Y_2{*}Y_3{*}\ldots{*}Y_h$ and tests for further redundancies, until no more redundancies are found. By contrast, if δ contains the configuration $Y_2{*}Y_3{*}\ldots{*}Y_h$ in combination with z, Y_1 makes a difference to Z and is, thus, not redundant. Accordingly, CNA re-adds Y_1 to $Y_2{*}Y_3{*}\ldots{*}Y_h$ and proceeds to eliminate Y_2, and so forth.

Similarly, to determine whether a complex necessary condition $\Phi_1 + \Phi_2 + \ldots + \Phi_h$ of a factor Z is minimally necessary, CNA systematically eliminates disjuncts from $\Phi_1 + \Phi_2 + \ldots + \Phi_h$ and checks for every resulting disjunction, say for $\Phi_2 + \Phi_3 + \ldots + \Phi_h$, whether it is still necessary for Z, i.e. whether δ contains a configuration featuring Z without any of the disjuncts in $\Phi_2 + \Phi_3 + \ldots + \Phi_h$. If δ does not contain such a configuration, $\Phi_2 + \Phi_3 + \ldots + \Phi_h$ is still necessary for Z, which means that the eliminated disjunct Φ_1 is redundant. Next, $\Phi_2 + \Phi_3 + \ldots + \Phi_h$ is tested for further redundancies, until no more redundancies are found.

CNA does not presuppose that certain factors in δ can be identified as endogenous prior to applying CNA. In principle, CNA is designed to recover and rigorously minimize all relationships of sufficiency and necessity among the factors in δ. In practice, however, it is often known from the outset which factors are exogenous and which endogenous. What is more, often enough theoretical knowledge is available to order the factors in δ causally, where a *causal ordering* is a relation $Y_i \prec Y_j$ entailing that Y_j cannot be a cause of Y_i (e.g., because Y_i is instantiated temporally before Y_j). That is, an ordering excludes certain causal dependencies but does not stipulate any. Accordingly, in addition to a data set δ, CNA may be given a subset W of endogenous factors (i.e. possible effects) in δ and an ordering $\prec$ over the factors in δ as input. Minimally sufficient and necessary conditions are then calculated for the members of W in accordance with $\prec$ only.

Recovered minimal theories of the elements of W are issued as so-called *atomic solution formulas*. If CNA finds an atomic solution formula $\Psi_1 \leftrightarrow Z_i$ and an atomic solution formula $\Psi_2 \leftrightarrow Z_j$ such that $Z_i \neq Z_j$ and Ψ_1 and Ψ_2 have at least one factor in common or Z_i appears in Ψ_2 or Z_j appears in Ψ_1, then CNA builds the *complex solution formula* $(\Psi_1 \leftrightarrow Z_i){*}(\Psi_2 \leftrightarrow Z_j)$. Configurational data regularly underdetermine their own causal modeling, with the effect that multiple atomic and complex solution formulas fit the data equally well. In cases of such model ambiguities, CNA provides an overview over the whole model space by returning all data-fitting solution formulas.

As causally analyzed data tend to be noisy, that is, confounded by uncontrolled (unmeasured) causes of endogenous factors, it often happens that no configuration of factors is strictly sufficient or necessary for a given $Z \in \mathsf{W}$. To still extract some (tentative) causal information from such data, Ragin (2006) has introduced so-called *consistency* and *coverage* measures (with values between 0 and 1). *Consistency* reproduces the degree to which the behavior of a given outcome obeys a corresponding sufficiency or necessity relationship (or a whole solution formula), whereas *coverage* reproduces the degree to which a sufficiency or necessity relationship (or a whole solution formula) accounts for the behavior of the corresponding outcome. If data cannot be causally modeled with maximal consistency and coverage scores, CNA invites its users to gradually lower consistency and coverage thresholds until solution formulas can be built.

The **cna** package by Ambuehl et al. (2015) implements the methodological protocol of CNA as sketched above. For more details on the background assumptions of CNA, its minimization algorithm, and its relation to other configurational methods such as QCA, we refer interested readers to Baumgartner (2009a).

Examples

Hypothetical data

In the following, we illustrate the main steps in using the **cna** package. First, we employ a hypothetical data set from Baumgartner (2009a) to investigate the causal dependencies among five factors hypothesized to constitute a causal structure behind the overall level of education in western democratic countries. These five factors are "strong unions" (*U*; 1 = strong, 0 = not strong), "high level of disparity" (*D*; 1 = high, 0 = not high), "strong left parties" (*L*; 1 = strong, 0 = not strong), "high gross national product" (GNP; *G*; 1 = high, 0 = not high) and "high level of education" (*E*; 1 = high, 0 = not high). The data for eight countries are presented in Table 1.

Case	*U*	*D*	*L*	*G*	*E*
1	1	1	1	1	1
2	1	1	1	0	1
3	1	0	1	1	1
4	1	0	1	0	1
5	0	1	1	1	1
6	0	1	1	0	1
7	0	0	0	1	1
8	0	0	0	0	0

Table 1: Exemplary data to be analyzed by CNA.

The **cna** package comes with an integrated bundle of six data sets from various areas of the social sciences. That bundle also includes the data in Table 1 as the data frame `d.educate`. Accordingly, the first step to causally model Table 1 by means of CNA is to load the **cna** package along with the `d.educate` data.

```
> library(cna)
> data(d.educate)
```

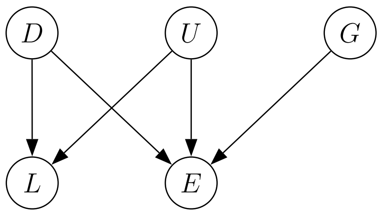
The heart of the **cna** package is constituted by the `cna()` function. It is the function that identifies and minimizes dependencies of sufficiency and necessity in the data, which can be given to `cna()` either in terms of a Boolean data frame or of a truth table as produced by the `truthTab()` function. Essentially, `truthTab()` simply merges multiple rows of a data frame featuring the same configuration into one row, such that each row of the resulting truth table corresponds to one determinate configuration. The number of occurrences (cases) and an enumeration of the cases are saved as attributes '`n`' and '`cases`', respectively. As Table 1 does not contain multiple rows with identical configurations, the application of `truthTab()` is uncalled for and we can directly pass `d.educate` on to `cna()`. Moreover, let us assume that we have no prior causal knowledge about the underlying causal structure, such that we cannot additionally supply a causal ordering. The following is the default output returned by `cna()`.

```
> cna(d.educate)
--- Coincidence Analysis (CNA) ---

Factors: U, D, L, G, E

Minimally sufficient conditions:
--------------------------------
Outcome D:
  condition consistency coverage
 L*u   -> D       1.000    0.500
 E*g*u -> D       1.000    0.250

Outcome E:
 condition consistency coverage
    L -> E       1.000    0.857
    D -> E       1.000    0.571
    G -> E       1.000    0.571
    U -> E       1.000    0.571
```

```
Outcome G:
  condition consistency coverage
 d*E*u -> G       1.000    0.250
 E*l   -> G       1.000    0.250

Outcome L:
 condition consistency coverage
  D   -> L       1.000    0.667
  U   -> L       1.000    0.667
  E*g -> L       1.000    0.500

Outcome U:
  condition consistency coverage
 d*L   -> U       1.000    0.500
 d*E*g -> U       1.000    0.250

Atomic solution formulas:
-------------------------
Outcome E:
       condition consistency coverage
 D + G + U <-> E       1.000    1.000
 G + L     <-> E       1.000    1.000

Outcome L:
   condition consistency coverage
 D + U <-> L       1.000    1.000

Complex solution formulas:
--------------------------
                           condition consistency coverage
 (D + G + U <-> E)  *  (D + U <-> L)       1.000    1.000
     (G + L <-> E)  *  (D + U <-> L)       1.000    1.000
```

First, cna() lists all minimally sufficient conditions of all factors in d.educate, second, it reports the atomic solution formulas for the factors that can be modeled as endogenous factors, and third, it specifies the resulting complex solutions. All solution types come with corresponding consistency and coverage scores. In case of Table 1, these scores reach maximal values for both atomic and complex solution formulas. Thus, the d.educate data are as good as configurational data can possibly get.

The above results show that the causal structure generating Table 1 features two endogenous factors, viz. "strong left parties" (*L*) and "high level of education" (*E*). Moreover, there is one atomic solution for *L* and there are two for *E*. Overall, cna() infers that the d.educate data can be modeled in terms of the two complex structures depicted in Figure 1. Graph 1a represents a common cause structure, in which "high level of disparity" (*D*) and "strong unions" (*U*) appear as direct common causes of *L* and *E*, whereas Graph 1b depicts a causal chain such that *D* and *U* are direct causes of *L*, which in turn is a direct cause of *E*. As the data in Table 1 are optimal by all standards of configurational modeling, there is no way to determine which of these two structures is the true or correct one.


(a) Common cause structure

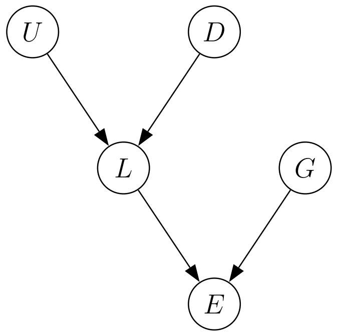

(b) Causal chain

Figure 1: Visualization of dependency structures in d.educate data.

Real-life data

This subsection illustrates further functionalities of the **cna** package on the basis of a real-life data set. To this end, we choose the study by Lam and Ostrom (2010), who analyze the effects of an irrigation experiment in the course of development interventions on the Indrawati River watershed in the central hills of Nepal. Among other things, the authors investigate the causal relevance of five exogenous factors on "persistent improvement in water adequacy at the tail end in winter" (W), which takes the value 1 when farmers at the tail end of the watershed persistently receive the water they need in winter, and the value 0 otherwise. The five exogenous factors are: "continual assistance on infrastructure improvement" (A), "existence of a set of formal rules for irrigation operation and maintenance" (R), "existence of provisions of fines" (F), "existence of consistent leadership" (L), and "existence of collective action among farmers for system maintenance" (C), for all of which the values 1 and 0 represent "yes" and "no", respectively. The relevant data set, which comprises 15 cases, is included in the **cna** package as the data frame `d.irrigate`.

```
> data(d.irrigate)
> d.irrigate
   A R F L C W
1  0 1 0 1 1 1
2  0 1 0 1 1 0
3  0 1 1 1 1 1
.. . . . . . .
<rest omitted>
```

Lam and Ostrom (2010) assume that W is the ultimate outcome of the data-generating causal structure. This background assumption can be given to `cna()` by means of the argument `ordering`, which takes a list of character vectors referring to the factors in the data frame as input. In case of `d.irrigate`, the intended ordering is this: `ordering = list(c("A","R","F","L"," C"),"W")`. It determines that W is causally located *after* A, R, F, L, C, meaning that the former cannot be a cause of the latter. Moreover, as this data frame does not comprise all relevant factors for W, it is no longer possible to reach perfect coverage scores. In the following analysis, we set the coverage threshold (`cov`) to 0.9 and extract only the complex solution formulas from the resulting solution object via the function `csf()`.

```
> sol1 <- cna(d.irrigate, ordering = list(c("A", "R", "F", "L", "C"), "W"), cov = 0.9)
> csf(sol1)
                                                 condition consistency coverage
1  (a + f*R + L <-> C)  *  (A*C + a*f*r + F*R + l*R <-> W)       1.000    0.917
2  (a + f*R + L <-> C)  *          (A*C + a*l + F*R <-> W)       1.000    0.917
3  (a + f*R + L <-> C)  *  (A*C + C*f*r + F*R + l*R <-> W)       1.000    0.917
4  (a + f*R + L <-> C)  *          (A*C + C*l + F*R <-> W)       1.000    0.917
5  (a + f*R + L <-> C)  *  (a*f*r + A*L + F*R + l*R <-> W)       1.000    0.917
6  (a + f*R + L <-> C)  *  (a*f*r + A*R + F*R + l*R <-> W)       1.000    0.917
7  (a + f*R + L <-> C)  *    (a*l + A*L + F*R + l*R <-> W)       1.000    0.917
8  (a + f*R + L <-> C)  *          (a*l + A*R + F*R <-> W)       1.000    0.917
9  (a + f*R + L <-> C)  *  (A*L + C*f*r + F*R + l*R <-> W)       1.000    0.917
10 (a + f*R + L <-> C)  *          (A*L + C*l + F*R <-> W)       1.000    0.917
11 (a + f*R + L <-> C)  *  (A*R + C*f*r + F*R + l*R <-> W)       1.000    0.917
12 (a + f*R + L <-> C)  *          (A*R + C*l + F*R <-> W)       1.000    0.917
```

This output of `cna()` shows that not only W can be modeled as an endogenous factor, but also C—a fact which is overlooked by Lam and Ostrom (2010) due to their reliance on QCA with its focus on single-outcome structures. `cna()` returns one atomic solution formula for C and 12 for W, yielding a total of 12 complex solution formulas that fare equally well with respect to all parameters of model fit. According to some of these models, the behavior of C and W is regulated by a common cause structure; according to others, the underlying structure is a causal chain.

To generate models for negative outcomes, `cna()` provides the argument `notcols`, which takes a character vector of factors to be negated as input. In the following analysis, we set `cov` to 0.66 and negate the factors C and W (which then must also appear negatively in the ordering). Moreover, we pass the solution object on to the `print()` function, which provides arguments determining the number of solutions to print (`nsolutions`) and what elements of the solution to print (`what`). The `what` argument takes a character vector as input, where `"t"` prints the truth table, `"m"` the minimally sufficient conditions, `"a"` the atomic solution formulas, `"c"` the complex solution formulas, and `"all"` returns all solution elements.

```
> sol2 <- cna(d.irrigate, ordering = list(c("A", "R", "F", "L", "c"), "w"),
              notcols = c("C", "W"),  cov = 0.66)
```

```
> print(sol2, nsolutions = 3, what = "a,c")
--- Coincidence Analysis (CNA) ---

Causal ordering:
A, R, F, L, c < w

Atomic solution formulas:
-------------------------
Outcome R:
             condition consistency coverage
 A*C + f*L       <-> R       1.000    0.667
 A*F + f*L       <-> R       1.000    0.667
 A*L + f*L + F*l <-> R       1.000    0.667

Outcome w:
       condition consistency coverage
 A*r + F*r <-> w       1.000    0.667
 A*r + L*r <-> w       1.000    0.667
 c*f + F*r <-> w       1.000    0.667
 ... (total no. of formulas: 6)

Complex solution formulas:
--------------------------
                                     condition consistency coverage
       (A*C + f*L <-> R)  *  (A*r + F*r <-> w)       1.000    0.667
       (A*F + f*L <-> R)  *  (A*r + F*r <-> w)       1.000    0.667
 (A*L + f*L + F*l <-> R)  *  (A*r + F*r <-> w)       1.000    0.667
 ... (total no. of formulas: 18)
```

Finally, the condition() function provides assistance to inspect the properties of sufficient and necessary conditions in a data frame, most notably, of minimally sufficient and necessary conditions that appear in solution formulas returned by cna(). It takes a vector of strings specifying Boolean functions as input, reveals which configurations and cases instantiate a given condition or solution, and lists consistency, coverage, as well as unique coverage scores (cf. Ragin, 2008, 63-68). Below, we investigate the properties of the first atomic solution for outcome w from the previous analysis.

```
> condition("A*r + F*r <-> w", d.irrigate)
A*r+F*r -> w :
 A*r+F*r w n cases
       0 0 1     1
       0 1 1     2
       0 0 2   3,4
       0 0 2   5,6
       0 0 2   7,8
       0 0 1     9
       0 0 1    10
       1 1 1    11
       1 1 1    12
       0 0 1    13
       0 0 1    14
       0 0 1    15
Consistency: 1.000 (2/2)
Coverage:    0.667 (2/3)
Total no. of cases: 15
Unique Coverages: A*r : 0.333 (1/3)
                  F*r : 0.333 (1/3)
```

The first two columns of the table returned by condition() indicate the configurations instantiating ('1') and not instantiating ('0') the disjunction $A{*}r + F{*}r$ and the outcome w, respectively. The third column specifies how many cases in the associated data feature a corresponding configuration, and the forth column lists these cases. According to the above output, hence, $A{*}r + F{*}r$ covers the instances of w in cases 11 and 12 and leaves the occurrence of w in case 2 uncovered. Consequently, the overall solution coverage is $2/3$, with each disjunct uniquely covering one of the instances of w.

Summary and outlook

We have presented **cna**, an R package implementing Coincidence Analysis (CNA), which is a method for the identification of multi-outcome structures in configurational data. CNA not only differs from QCA—the standard method of configurational causal modeling—by relaxing the single-outcome restriction but also by not drawing on Quine-McCluskey optimization for the elimination of redundancies from sufficient and necessary conditions. Instead, CNA employs its own minimization algorithm that is custom-built for causal modeling purposes.

At this stage of development, **cna** still requires bivalent variables. Planned future enhancements include the capability to process multivalent factors that generate crisp sets (Thiem, 2013) and bivalent factors with fuzzy sets (Smithson and Verkuilen, 2006). Possibilities to merge these constructs in multivalent factors with fuzzy sets, as has recently been suggested in the context of QCA (Thiem, 2014a), will be explored as well. In this connection, aspects of alternative procedures proposed in the context of minimization with fuzzy sets may be incorporated where appropriate (Eliason and Stryker, 2009). Finally, functionality for sensitivity diagnostics that facilitates robustness tests is envisaged (Thiem, 2014b).

Complex causal structures are communicated most effectively to readers of scientific articles in the form of graphs rather than formulas. This is all the more true for multivalent factors. In this regard, functionality that translates **cna** solutions into corresponding graphs enjoys high priority on the list of future enhancements.

Acknowledgment

This work was generously supported by the Swiss National Science Foundation, grant number PP00P1_144736.

Bibliography

M. Ambuehl, M. Baumgartner, R. Epple, A. Kauffmann, and A. Thiem. *cna: A Package for Coincidence Analysis (CNA)*, 2015. URL http://CRAN.R-project.org/package=cna. R Package Version 1.0-3. [p176, 177]

M. Baumgartner. Regularity theories reassessed. *Philosophia*, 36(3):327–354, 2008. [p176]

M. Baumgartner. Inferring causal complexity. *Sociological Methods & Research*, 38(1):71–101, 2009a. [p176, 177, 178]

M. Baumgartner. Uncovering deterministic causal structures: A Boolean approach. *Synthese*, 170(1): 71–96, 2009b. [p176]

M. Baumgartner. Detecting causal chains in small-n data. *Field Methods*, 25(1):3–24, 2013a. [p176]

M. Baumgartner. A regularity theoretic approach to actual causation. *Erkenntnis*, 78:85–109, 2013b. [p176]

M. Baumgartner and A. Thiem. When there is more than meets the eye: Model ambiguities in configurational comparative research. Paper presented at the *2nd International QCA Expert Workshop*, ETH Zürich, 2014. [p176]

C.-N. Chung. Markets, culture and institutions: The emergence of large business groups in Taiwan, 1950s–1970s. *Journal of Management Studies*, 38(5):719–745, 2001. [p176]

D. Cragun, R. D. DeBate, S. T. Vadaparampil, J. Baldwin, H. Hampel, and T. Pal. Comparing universal Lynch syndrome tumor-screening programs to evaluate associations between implementation strategies and patient follow-through. *Genetics in Medicine*, 16(10):773–782, 2014. [p176]

L. Cronqvist. *Tosmana: Tool for Small-n Analysis, Version 1.3.2.0 [computer program]*. University of Trier, Trier, 2011. [p176]

M. Crowley. Gender, the labor process and dignity at work. *Social Forces*, 91(4):1209–1238, 2013. [p176]

A. Dusa and A. Thiem. *Qualitative Comparative Analysis*, 2014. URL http://CRAN.R-project.org/package=QCA. R Package Version 1.1-4. [p176]

S. R. Eliason and R. Stryker. Goodness-of-fit tests and descriptive measures in fuzzy-set analysis. *Sociological Methods & Research*, 38(1):102–146, 2009. [p182]

G. Graßhoff and M. May. Causal regularities. In W. Spohn, M. Ledwig, and M. Esfeld, editors, *Current issues in causation*, pages 85–114. Mentis, Paderborn, 2001. [p176]

R. Huang. *QCA3: Yet Another Package for Qualitative Comparative Analysis*, 2014. URL `http://CRAN.R-project.org/package=QCA3`. R package version 0.0-7. [p176]

W. F. Lam and E. Ostrom. Analyzing the dynamic complexity of development interventions: Lessons from an irrigation experiment in Nepal. *Policy Sciences*, 43(1):1–25, 2010. [p180]

K. C. Longest and P. A. Thoits. Gender, the stress process, and health: A configurational approach. *Society and Mental Health*, 2(3):187–206, 2012. [p176]

K. C. Longest and S. Vaisey. fuzzy: A program for performing Qualitative Comparative Analyses (QCA) in Stata. *Stata Journal*, 8(1):79–104, 2008. [p176]

J. L. Mackie. *The Cement of the Universe: A Study of Causation*. Clarendon Press, Oxford, 1974. [p176]

E. J. McCluskey. *Introduction to the Theory of Wwitching Circuits*. Princeton University Press, Princeton, 1965. [p176]

M. Quaranta. *SetMethods: A package companion to "Set-theoretic methods for the social sciences"*, 2013. URL `http://CRAN.R-project.org/package=SetMethods`. R package version 1.0. [p176]

W. v. O. Quine. On cores and prime implicants of truth functions. *The American Mathematical Monthly*, 66(9):755–760, 1959. [p176]

C. C. Ragin. *The Comparative Method: Moving Beyond Qualitative and Quantitative Strategies*. University of California Press, Berkeley, 1987. [p176]

C. C. Ragin. *Fuzzy-Set Social Science*. University of Chicago Press, Chicago, 2000. [p176]

C. C. Ragin. Set relations in social research: Evaluating their consistency and coverage. *Political Analysis*, 14(3):291–310, 2006. [p177]

C. C. Ragin. *Redesigning Social Inquiry: Fuzzy Sets and Beyond*. University of Chicago Press, Chicago, 2008. [p176, 181]

C. C. Ragin and S. Davey. *fs/QCA: Fuzzy-set/Qualitative Comparative Analysis, Version 2.5 [computer program]*. Department of Sociology, University of California, Irvine, 2014. [p176]

C. Reichert and C. Rubinson. *Kirq, Version 2.1.12 [computer program]*. University of Houston-Downtown, Houston, 2014. [p176]

B. Rihoux and C. C. Ragin, editors. *Configurational Comparative Methods: Qualitative Comparative Analysis (QCA) and Related Techniques*. Sage, London, 2009. [p176]

C. Q. Schneider and C. Wagemann. *Set-Theoretic Methods for the Social Sciences: A Guide to Qualitative Comparative Analysis (QCA)*. Cambridge University Press, Cambridge, 2012. [p176]

M. Smithson and J. Verkuilen. *Fuzzy Set Theory: Applications in the Social Sciences*. Sage, London, 2006. [p182]

A. Thiem. Conditions of intergovernmental armaments cooperation in Western Europe, 1996–2006. *European Political Science Review*, 3(1):1–33, 2011. [p176]

A. Thiem. Clearly crisp, and not fuzzy: A reassessment of the (putative) pitfalls of multi-value QCA. *Field Methods*, 25(2):197–207, 2013. [p182]

A. Thiem. Unifying configurational comparative methods: Generalized-set Qualitative Comparative Analysis. *Sociological Methods & Research*, 43(2):313–337, 2014a. [p182]

A. Thiem. Membership function sensitivity of descriptive statistics in fuzzy-set relations. *International Journal of Social Research Methodology*, 17(6):625–642, 2014b. [p182]

A. Thiem. Navigating the complexities of Qualitative Comparative Analysis: Case numbers, necessity relations, and model ambiguities. *Evaluation Review*, 38(6):487–513, 2014c. [p176]

A. Thiem and A. Duşa. QCA: A package for Qualitative Comparative Analysis. *The R Journal*, 5(1): 87–97, 2013a. [p176]

A. Thiem and A. Duşa. Boolean minimization in social science research: A review of current software for Qualitative Comparative Analysis (QCA). *Social Science Computer Review*, 31(4):505–521, 2013b. [p176]

A. Thiem and A. Duşa. *Qualitative Comparative Analysis with R: A User's Guide*. Springer, New York, 2013c. [p176]

N. van Vliet, A. Reenberg, and L. V. Rasmussen. Scientific documentation of crop land changes in the Sahel: A half empty box of knowledge to support policy? *Journal of Arid Environments*, 95:1–13, 2013. [p176]

Michael Baumgartner
Department of Philosophy
University of Geneva
Rue de Candolle 2 / Bât. Landolt
1211 Geneva
Switzerland
`michael.baumgartner@unige.ch`

Alrik Thiem
Department of Philosophy
University of Geneva
Rue de Candolle 2 / Bât. Landolt
1211 Geneva
Switzerland
`alrik.thiem@unige.ch`

Manipulation of Discrete Random Variables with discreteRV

by Eric Hare, Andreas Buja and Heike Hofmann

Abstract A prominent issue in statistics education is the sometimes large disparity between the theoretical and the computational coursework. **discreteRV** is an R package for manipulation of discrete random variables which uses clean and familiar syntax similar to the mathematical notation in introductory probability courses. The package offers functions that are simple enough for users with little experience with statistical programming, but has more advanced features which are suitable for a large number of more complex applications. In this paper, we introduce and motivate **discreteRV**, describe its functionality, and provide reproducible examples illustrating its use.

Introduction

One of the primary hurdles in teaching probability courses in an undergraduate setting is to bridge the gap between theoretical notation from textbooks and lectures, and the statements used in statistical software required in more and more classes. Depending on the background of the student, this missing link can manifest itself differently: some students master theoretical concepts and notation, but struggle with the computing environment, while others are very comfortable with statistical programming, but find it difficult to translate their knowledge back to the theoretical setting of the classroom.

discreteRV (Buja et al., 2015) is an approach to help with bringing software commands closer to the theoretical notation. The package provides a comprehensive set of functions to create, manipulate, and simulate from discrete random variables. It is designed for introductory probability courses. **discreteRV** uses syntax that closely matches the notation of standard probability textbooks to allow for a more seamless connection between a probability classroom setting and the use of statistical software. **discreteRV** is available for download on the Comprehensive R Archive Network (CRAN). **discreteRV** was derived from a script written by Dr. Andreas Buja for an introductory statistics class (Buja). The package **rv2** (Buja and Wickham, 2014), available on GitHub, provides a useful example of using **devtools** (Wickham and Chang, 2015) to begin basic package development, and also uses Dr. Buja's code as a starting point. The goal of **rv2** seems more focused on learning package development, while the goal of **discreteRV** is to be a useful statistics education and probability learning tool.

The functions of **discreteRV** are organized into two logical areas, termed probabilities and simulations. This document will illustrate the use of both sets of functions. All code used in this document is available in a vignette, accessible by loading **discreteRV** and calling `vignette("discreteRV")`.

Probabilities

discreteRV includes a suite of functions to create, manipulate, and compute distributional quantities for discrete random variables. A list of these functions and brief discriptions of their functionality is available in Table 1.

Creating random variables

The centerpiece of **discreteRV** is a set of functions to create and manipulate discrete random variables. A random variable X is defined as a theoretical construct representing the value of an outcome of a random experiment (see e.g. Wild and Seber, 1999). A discrete random variable is a special case that can only take on a countable set of values. Discrete random variables are associated with probability mass functions, which map the set of possible values of the random variable to probabilities. Probability mass functions must therefore define probabilities which are between zero and one, and must sum to one.

Throughout this document, we will work with two random variables, a simple example of a discrete random variable representing the value of a roll of a fair die, and one representing a realization of a Poisson random variable with mean parameter equal to two. Formally, we can define such random variables and their probability mass functions as follows:

Let X be a random variable representing a single roll of a fair die; i.e., the sample space $\Omega = \{1, 2, 3, 4, 5, 6\}$ and X is the identity, mapping the upturned face of a die roll to the corresponding number of dots visible. Then,

Name	Description
Creation	
`RV`	Create a random variable consisting of possible outcome values and their probabilities or odds
`as.RV`	Turn a probability vector with possible outcome values in the `names()` attribute into a random variable
`jointRV`	Create a joint random variable consisting of possible outcome values and their probabilities or odds
Manipulation	
`iid`	Returns a random variable with joint probability mass function of random variable X^n
`independent`	Returns a boolean indicating whether two RVs X and Y are independent
`joint`	Returns a random variable with joint probability mass function of random variables X and Y
`marginal`	The specified marginal distribution of a joint random variable
`margins`	All marginal distributions of a joint random variable
`SofI`	Sum of independent random variables
`SofIID`	Sum of independent identically distributed random variables
Probabilities	
`P`	Calculate probabilities of events
`probs`	Probability mass function of random variable X
`E`	Expected value of a random variable
`V`	Variance of a random variable
`SD`	Standard deviation of a random variable
`SKEW`	Skewness of a random variable
`KURT`	Kurtosis of a random variable
Methods for `"RV"` objects	
`plot`	Plot a random variable of class `"RV"`
`print`	Print a random variable of class `"RV"`
`qqnorm`	Normal quantile plot for `"RV"` objects to answer the question how close to normal it is

Table 1: Overview of functions provided in **discreteRV** ordered by topics.

$$f(x) = P(X = x) = \begin{cases} 1/6 & x \in \{1,2,3,4,5,6\} \\ 0 & \text{otherwise} \end{cases}$$

Let Y be a random variable distributed according to a Poisson distribution with mean parameter λ. In this case, Y takes on values in the non-negative integers $\{0,1,2,\ldots\}$. Then,

$$f(y) = P(Y = y) = \begin{cases} \frac{\lambda^y e^{-\lambda}}{y!} & y \in \{0,1,2,\ldots\} \\ 0 & \text{otherwise} \end{cases}$$

In **discreteRV**, a discrete random variable is defined through the use of the `RV` function. `RV` accepts a vector of outcomes, a vector of probabilities, and returns an `"RV"` object. The code to create `X`, a random variable representing the roll of a fair die, is as follows:

```
> (X <- RV(outcomes = 1:6, probs = 1/6))

Random variable with 6 outcomes

Outcomes   1   2   3   4   5   6
Probs    1/6 1/6 1/6 1/6 1/6 1/6
```

Defaults are chosen to simplify the random variable creation process. For instance, if the `probs` argument is left unspecified, **discreteRV** assumes a uniform distribution. Hence, the following code is equivalent for defining a fair die:

```
> (X <- RV(1:6))
```

discreteRV	Casella and Berger
`E(X)`	E(X)
`P(X == x)`	$P(X = x)$
`P(X >= x)`	$P(X \geq x)$
`P((X < x1) %AND% (X > x2))`	$P(X < x_1 \cap X > x_2)$
`P((X < x1) %OR% (X > x2))`	$P(X < x_1 \cup X > x_2)$
`P((X == x1) | (X > x2))`	$P(X < x_1 \| X > x_2)$
`probs(X)`	$f(x)$
`V(X)`	$Var(X)$

Table 2: Probability functions in **discreteRV** and their corresponding syntax in introductory statistics courses.

```
Random variable with 6 outcomes

Outcomes   1   2   3   4   5   6
Probs    1/6 1/6 1/6 1/6 1/6 1/6
```

Outcomes can be specified as a range of values, which is useful for distributions in which the outcomes that can occur with non-zero probability are unbounded. This can be indicated with the `range` argument, which defaults to `TRUE` in the event that the range of values includes positive or negative infinity. To define our Poisson random variable Y, we specify the outcomes as a range and the probabilities as a function:

```
> pois.func <- function(y, lambda) { return(lambda^y * exp(-lambda) / factorial(y)) }
> (Y <- RV(outcomes = c(0, Inf), probs = pois.func, lambda = 2))

Random variable with outcomes from 0 to Inf

Outcomes     0     1     2     3     4     5     6     7     8     9    10    11
Probs    0.135 0.271 0.271 0.180 0.090 0.036 0.012 0.003 0.001 0.000 0.000 0.000

Displaying first 12 outcomes
```

Several common distributions are natively supported so that the functions need not be defined manually. For instance, an equivalent method of defining Y is:

```
> (Y <- RV("poisson", lambda = 2))

Random variable with outcomes from 0 to Inf

Outcomes     0     1     2     3     4     5     6     7     8     9    10    11
Probs    0.135 0.271 0.271 0.180 0.090 0.036 0.012 0.003 0.001 0.000 0.000 0.000

Displaying first 12 outcomes
```

The `RV` function also allows the definition of a random variable in terms of odds. We construct a loaded die in which a roll of one is four times as likely as any other roll as:

```
> (X.loaded <- RV(outcomes = 1:6, odds = c(4, 1, 1, 1, 1, 1)))

Random variable with 6 outcomes

Outcomes   1   2   3   4   5   6
Odds     4:5 1:8 1:8 1:8 1:8 1:8
```

Structure of an "RV" object

The syntactic structure of the included functions lends itself both to a natural presentation of elementary probabilities and properties of probability mass functions in an introductory probability course, as well as more advanced modeling of discrete random variables. In Table 2, we provide an overview of the notational similarities between **discreteRV** and the commonly used probability textbook by Casella and Berger (2001).

A random variable object is constructed by defining a standard R vector to be the possible outcomes that the random variable can take (the sample space Ω). It is preferred, though not required, that these be encoded as numeric values, since this allows the computation of expected values, variances, and other distributional properties. This vector of outcomes then stores attributes which include the probability of each outcome. By default, the print method for a random variable will display the probabilities as fractions in most cases, aiding in readability. The probabilities can be retrieved as a numeric vector by using the `probs` function:

```
> probs(X)

        1         2         3         4         5         6
0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667
```

Probability-based calculations

By storing the outcomes as the principal component of the object `X`, we can make a number of probability statements in R. For instance, we can calculate the probability of obtaining a roll greater than 1 by using the code $P(X > 1)$. R will check which of the values in the vector `X` are greater than 1. In this case, these are the outcomes 2, 3, 4, 5, and 6. Hence, R will return `TRUE` for these elements of `X`, and we compute the probability of this occurrence in the function `P` by simply summing over the probability values stored in the names of these particular outcomes. Likewise, we can make slightly more complicated probability statements such as $P(X > 5 \cup X = 1)$, using the `%OR%` and `%AND%` operators. Consider our Poisson random variable Y, and suppose we want to obtain the probability that Y is within a distance δ of its mean parameter $\lambda = 2$:

```
> delta <- 3; lambda <- 2
> P((Y >= lambda - delta) %AND% (Y <= lambda + delta))

[1] 0.9834364
```

Alternatively, we could have also used the slightly more complicated looking expression:

```
> P((Y - lambda)^2 <= delta^2)

[1] 0.9834364
```

Conditional probabilities often provide a massive hurdle for students of introductory probability classes. These types of questions often make it necessary to first translate the problem from everyday language into the mathematical concept of conditional probability, e.g., what is the probability that you will not need an umbrella when the weather forecast said it was not going to rain? Similarly, what is the probability that a die shows a one, if we already know that the roll is no more than 3? The mathematical solution is, of course, $P(X = 1 \mid X \leq 3)$. In **discreteRV** this translates directly to a solution of `P(X == 1 | X <= 3)`. The use of the pipe operator may be less intuitive to the seasoned R programmer, but overcomes a major notational issue in that conditional probabilities are most commonly specified with the pipe. Using the pipe for conditional probablity, we had to create alternative `%OR%` and `%AND%` operators, as specified previously.

We can compute several other distributional quantities, including the expected value and the variance of a random variable. In notation from probability courses, expected values can be found with the `E` function. To compute the expected value for a single roll of a fair die, we run the code `E(X)`. The expected value for a Poisson random variable is its mean, and hence `E(Y)` in our example will return the value two. The function `V(X)` computes the variance of random variable X. Alternatively, we can also work from first principles and assure ourselves that the expression `E((X -E(X))ˆ2)` provides the same result:

```
> V(X)

[1] 2.916667

> E((X - E(X))^2)

[1] 2.916667
```

Joint distributions

Aside from moments and probability statements, **discreteRV** includes a powerful set of functions used to create joint probability distributions. Once again letting X be a random variable representing a

Outcome	1,1	1,2	1,3	1,4	1,5	1,6	2,1	2,2
Probability	1/36	1/36	1/36	1/36	1/36	1/36	1/36	1/36

Table 3: First eight outcomes and their associated probabilities for a variable representing two independent rolls of a die.

single die roll, we can use the `iid` function to compute the probability mass function of n trials of X. Table 3 gives the first eight outcomes for $n = 2$, and Table 4 gives the first eight outcomes for $n = 3$. Notice again that the probabilities have been coerced into fractions for readability. Notice also that the outcomes of the joint distribution are encoded by the outcomes on each trial separated by a comma.

Outcome	1,1,1	1,1,2	1,1,3	1,1,4	1,1,5	1,1,6	1,2,1	1,2,2
Probability	1/216	1/216	1/216	1/216	1/216	1/216	1/216	1/216

Table 4: First eight outcomes and their associated probabilities for a variable representing three independent rolls of a die.

The `*` operator has been overloaded in order to allow a more seamless syntax for defining joint distributions. Suppose we wish to compute the joint distribution of `X`, our toss of a fair coin, and a coin flip. After defining the coin flip variable, the joint distribution can be defined as follows:

```
> Z <- RV(0:1)
> X * Z

Random variable with 12 outcomes

Outcomes  1,0  1,1  2,0  2,1  3,0  3,1  4,0  4,1  5,0  5,1  6,0  6,1
Probs    1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12
```

Note that the behavior is slightly different when using the `*` operator on the same random variable. That is, `X * X` will not compute a joint distribution of two realizations of X, but will rather return the random variable with the original outcomes squared, and the same probabilities. This allows us to perform computations such as `E(X^2)` without encountering unexpected behavior.

Joint distributions need not be the product of iid random variables. Joint distributions in which the marginal distributions are dependent can also be defined. Consider the probability distribution defined in Table 5. Note that A and B are dependent, as the product of the marginal distributions does not equal the joint. We can define such a random variable in **discreteRV** by using the `jointRV` function, which is a wrapper for `RV`:

```
> (AandB <- jointRV(outcomes = list(1:3, 0:2), probs = 1:9 / sum(1:9)))

Random variable with 9 outcomes

Outcomes  1,0  1,1  1,2  2,0  2,1  2,2  3,0  3,1  3,2
Probs    1/45 2/45 1/15 4/45  1/9 2/15 7/45 8/45  1/5
```

The individual marginal distributions can be obtained by use of the `marginal` function:

```
> A <- marginal(AandB, 1)
> B <- marginal(AandB, 2)
```

Although the marginal distributions allow all the same computations of any univariate random variable, they maintain a special property. The joint distribution that produced the marginals is stored as attributes in the object. This allows for several more advanced probability calculations, involving the marginal and conditional distributions:

```
> P(A < B)

[1] 0.06666667

> P(A == 2 | B > 0)

[1] 0.3333333

> P(A == 2 | (B == 1) %OR% (B == 2))
```

	1	2	3
0	1/45	4/45	7/45
1	2/45	1/9	8/45
2	1/15	2/15	1/5

Table 5: Outcomes and their associated probabilities for a joint distribution of random variables A (along the columns) and B (along the rows).

Outcome	2	3	4	5	6	7	8	9	10	11	12
Probability	1/36	1/18	1/12	1/9	5/36	1/6	5/36	1/9	1/12	1/18	1/36

Table 6: Outcomes and their associated probabilities for a variable representing the sum of two independent rolls of a die.

```
[1] 0.3333333

> independent(A, B)

[1] FALSE

> A | (A > 1)

Random variable with 2 outcomes

Outcomes    2    3
Probs    5/13 8/13

> A | (B == 2)

Random variable with 3 outcomes

Outcomes   1   2   3
Probs    1/6 1/3 1/2

> E(A | (B == 2))

[1] 2.333333
```

discreteRV also includes functions to compute the sum of independent random variables. If the variables are identically distributed, the `SofIID` function can be used to compute probabilities for the sum of n independent realizations of the random variable. In our fair die example, `SofIID(X,2)` creates a random variable object for the sum of two fair dice as shown in Table 6.

The `SofI` function computes the random variable representing the sum of two independent, but not necessarily identically distributed, random variables. The + operator is overloaded to make this computation even more syntactically friendly. Note, however, that similar limitations apply as in the joint distribution case:

```
> X + Z

Random variable with 7 outcomes

Outcomes    1    2    3    4    5    6    7
Probs    1/12  1/6  1/6  1/6  1/6  1/6 1/12

> X + X # Note that this is NOT a random variable for X1 + X2

Random variable with 6 outcomes

Outcomes   2   4   6   8  10  12
Probs    1/6 1/6 1/6 1/6 1/6 1/6

> 2 * X # Same as above

Random variable with 6 outcomes

Outcomes   2   4   6   8  10  12
Probs    1/6 1/6 1/6 1/6 1/6 1/6
```

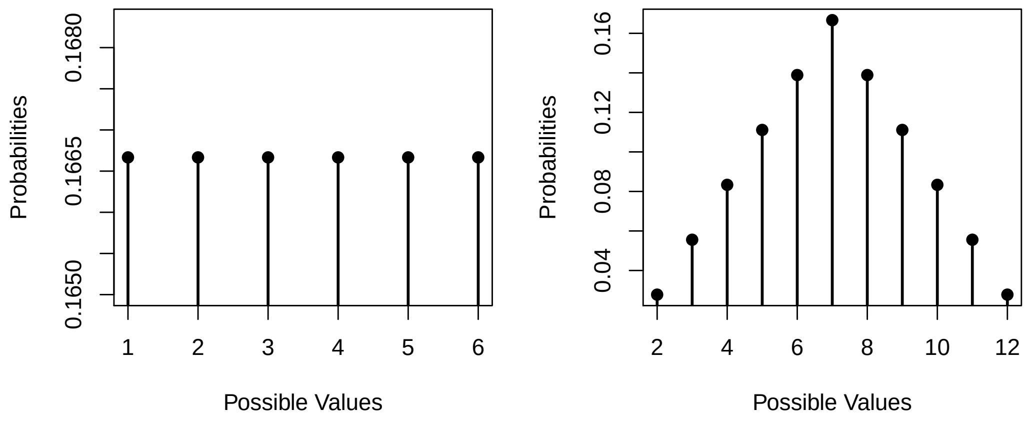


Figure 1: Left: plot method called on a fair die random variable; right: plot method called on a sum of two fair die random variables.

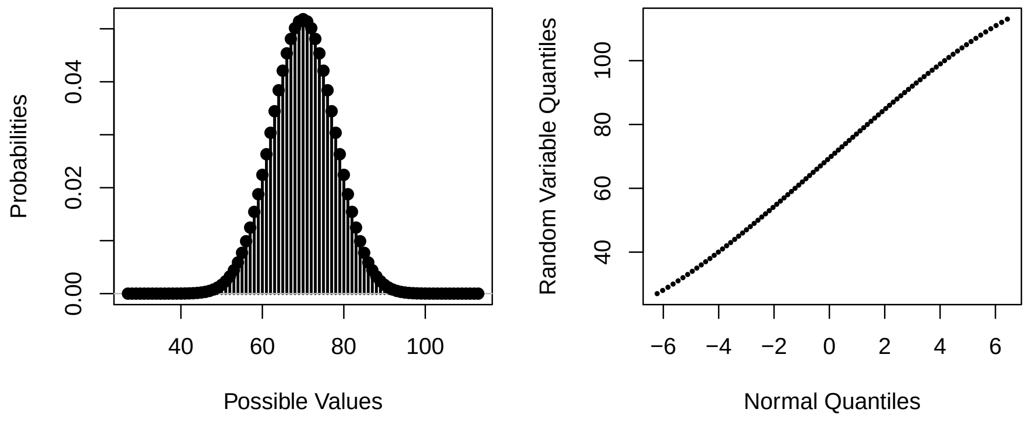


Figure 2: Left: plot method called on a sum of 20 fair die random variables; right: `qqnorm` method called on a sum of 20 fair die random variables.

Plotting

discreteRV includes a `plot` method for random variable objects so that visualizing outcomes and their probabilities is as simple as calling `plot(X)`. Figure 1 on the left shows a visual representation of the probability mass function (pmf) of a fair die. The x axis includes all outcomes, and the y axis includes the probabilities of each particular outcome. Figure 1 on the right shows the pmf of the sum of two independent rolls of a fair die. The pmf of a sum of 20 independent rolls of a die is given in Figure 2 on the left.

In addition to a plotting method, there is also a method for `qqnorm` to allow assessment of normality for random variable objects, as displayed in Figure 2 on the right. While very close to a normal, the sum of 20 independent rolls of a fair die still shows a slight S curve in the Q-Q plot.

Simulation

discreteRV also includes a set of functions to simulate trials from a random variable. A list of these functions and brief discriptions of their functionality is available in Table 7.

Creation

Creating a simulated random vector is done by using the `rsim` function. `rsim` accepts a parameter X representing the random variable to simulate from, and a parameter n representing the number of independent trials to simulate. For example, suppose we would like to simulate ten trials from a fair die. We have already created a random variable object `X`, so we simply call `rsim` as follows:

Name	Description
`plot`	Plot method for a simulated random vector, i.e., an object of class `"RVsim"`
`Prop`	Proportion of an event observed in a vector of simulated trials
`props`	Proportions of observed outcomes in one or more vectors of simulated trials
`rsim`	Simulate n independent trials from a random variable X
`skewSim`	Skew of the empirical distribution of simulated data

Table 7: List of the simulation functions contained in **discreteRV**.

```
> (X.sim <- rsim(X, 10))

Simulated Vector:  4 3 2 3 4 2 2 3 6 5

Random variable with 6 outcomes

Outcomes   1   2   3   4   5   6
Probs    1/6 1/6 1/6 1/6 1/6 1/6
```

The object returned is a vector of simulated values, with an attribute containing the random variable that was used for the simulation. If we would like to retrieve only the simulated values and exclude the attached probabilities, we can coerce the object into a vector using R's built-in `as.vector` function.

```
> as.vector(X.sim)

 [1] 4 3 2 3 4 2 2 3 6 5
```

It is also possible to retrieve some quantities from the simulation. We can retrieve the empirical distribution of simulated values with the `props` function. This will return the outcomes from the original random variable object, and the observed proportion of simulated values for each of the outcomes. We can also compute observed proportions of events by using the `Prop` function. Similar to the `P` function for probability computations on random variable objects, `Prop` accepts a variety of logical statements.

```
> props(X.sim)

RV
  1   2   3   4   5   6
0.0 0.3 0.3 0.2 0.1 0.1

> Prop(X.sim == 3)

[1] 0.3

> Prop(X.sim > 3)

[1] 0.4
```

Extended example: playing Craps

Craps is a common dice game played in casinos. The game begins with what is called the "Come Out" roll, in which two fair dice are rolled. If a sum of seven or eleven is obtained, the player wins. If a sum of two, three, or twelve is obtained, the player loses. In all other cases, the roll obtained is declared the "Point" and the player rolls again in an attempt to obtain this same point value. If the player rolls the Point, they win, but if they roll a seven, they lose. Rolls continue until one of these two outcomes is achieved.

discreteRV allows for a seamless analysis and simulation of the probabilities associated with different events in Craps. Let us begin by asking "What is the probability that the game ends after the first roll?" To answer this question we construct two random variables. We note that calling `RV(1:6)` returns a random variable for a single roll of a fair die, and then we use the overloaded `+` operator to sum over two rolls to obtain the random variable `Roll`.

```
> (Roll <- RV(1:6) + RV(1:6))
```

```
Random variable with 11 outcomes

Outcomes     2    3    4    5    6    7    8    9   10   11   12
Probs     1/36 1/18 1/12  1/9 5/36  1/6 5/36  1/9 1/12 1/18 1/36
```

Recall that the game ends after the first roll if and only if a seven or eleven is obtained (resulting in a win), or a two, three, or twelve is obtained (resulting in a loss). Hence, we calculate the probability that the game ends after the first roll as follows:

```
> P(Roll %in% c(7, 11, 2, 3, 12))

[1] 0.3333333
```

Now suppose we would like to condition on the game having ended after the first roll. Using the conditional probability operator in **discreteRV**, we can obtain the probabilities of winning and losing given that the game ended after the first roll:

```
> P(Roll %in% c(7, 11) | Roll %in% c(7, 11, 2, 3, 12))

[1] 0.6666667

> P(Roll %in% c(2, 3, 12) | Roll %in% c(7, 11, 2, 3, 12))

[1] 0.3333333
```

Now, let us turn our attention to calculating the probability of winning a game in two rolls. Recall that we can use the `iid` function to generate joint distributions of independent and identically distributed random variables. In this case, we would like to generate the joint distribution for two independent rolls of two dice. Now, we will have 11^2 possible outcomes, and our job is to determine which outcomes result in a win. We know that any time the first roll is a seven or eleven, we will have won. We also know that if the roll is between four and ten inclusive, then we will get to roll again. To win within two rolls given that we have received a four through ten requires that the second roll matches the first. We can enumerate the various possibilities to calculate the probability of winning in two rolls, which is approximately 30%.

```
> TwoRolls <- iid(Roll, 2)
> First <- marginal(TwoRolls, 1)
> Second <- marginal(TwoRolls, 2)
> P(First %in% c(7, 11) %OR% (First %in% 4:10 %AND% (First == Second)))

[1] 0.2993827
```

Finally, suppose we are interested in the empirical probability of winning a game of Craps. Using the simulation functions in **discreteRV**, we can write a routine to simulate playing Craps. Using the `rsim` function, we simulate a single game of Craps by rolling from our random variable `Roll`, which represents the sum of two dice. We then perform this simulation 100000 times. The results indicate that the player wins a game of craps approximately 49% of the time.

```
> craps_game <- function(RV) {
+
+    my.roll <- rsim(RV, 1)
+
+    if (my.roll %in% c(7, 11)) { return(1) }
+    else if (my.roll %in% c(2, 3, 12)) { return(0) }
+    else {
+        new.roll <- 0
+        while (new.roll != my.roll & new.roll != 7) {
+           new.roll <- rsim(RV, 1)
+        }
+
+        return(as.numeric(new.roll == my.roll))
+    }
+ }
> sim.results <- replicate(100000, craps_game(Roll))
> mean(sim.results)

[1] 0.4944
```

Conclusion

The power of **discreteRV** is truly in its simplicity. Because it uses familiar introductory probability syntax, it allows students who may not be experienced or comfortable with programming to ease into computer-based computations. Nonetheless, **discreteRV** also includes several powerful functions for analyzing, summing, and combining discrete random variables which can be of use to the experienced programmer.

Bibliography

A. Buja. *Basic Probability in R*. URL http://stat.wharton.upenn.edu/~buja/STAT-101/src-probability.R. [p185]

A. Buja and H. Wickham. *rv2: Discrete Random Variables*, 2014. URL https://github.com/hadley/rv2. R package version 0.1. [p185]

A. Buja, E. Hare, and H. Hofmann. *discreteRV: Create and Manipulate Discrete Random Variables*, 2015. URL http://CRAN.R-project.org/package=discreteRV. R package version 1.2.1. [p185]

G. Casella and R. L. Berger. *Statistical Inference*, volume 2. Cengage Learning, 2001. [p187]

H. Wickham and W. Chang. *devtools: Tools to Make Developing R Packages Easier*, 2015. URL http://CRAN.R-project.org/package=devtools. R package version 1.7.0. [p185]

C. Wild and G. Seber. *Chance Encounters: A First Course in Data Analysis and Inference*. John Wiley & Sons, 1999. [p185]

Eric Hare
Iowa State University
1121 Snedecor Hall
Ames, IA, 50011
erichare@iastate.edu

Andreas Buja
University of Pennsylvania
400 Jon M. Huntsman Hall
Philadelphia, PA, 19104
buja.at.wharton@gmail.com

Heike Hofmann
Iowa State University
2413 Snedecor Hall
Ames, IA, 50011
hofmann@iastate.edu

Estimability Tools for Package Developers

by Russell V. Lenth

Abstract When a linear model is rank-deficient, then predictions based on that model become questionable because not all predictions are uniquely estimable. However, some of them are, and the **estimability** package provides tools that package developers can use to tell which is which. With the use of these tools, a model object's `predict` method could return estimable predictions as-is while flagging non-estimable ones in some way, so that the user can know which predictions to believe. The **estimability** package also provides, as a demonstration, an estimability-enhanced `epredict` method to use in place of `predict` for models fitted using the **stats** package.

Introduction

Consider a linear regression or mixed model having fixed component of the matrix form $\mathbf{X}\boldsymbol{\beta}$. If $\mathbf{X}$ is not of full column rank, then there is not a unique estimate $\mathbf{b}$ of $\boldsymbol{\beta}$. However, consider using $\boldsymbol{\lambda}'\mathbf{b}$ to estimate the value of some linear function $\boldsymbol{\lambda}'\boldsymbol{\beta} = \sum_j \lambda_j \beta_j$. (We use $\mathbf{x}'$ to denote the transpose of a vector $\mathbf{x}$.) For some $\boldsymbol{\lambda}$s, the prediction depends on the solution $\mathbf{b}$; but for others—the estimable ones—it does not.

An illustration

An example will help illustrate the issues. In the following commands, we create four predictors `x1`–`x4` and a response variable `y`:

```
> x1 <- -4 : 4
> x2 <- c(-2, 1, -1, 2, 0, 2, -1, 1,-2)
> x3 <- 3 * x1 - 2 * x2
> x4 <- x2 - x1 + 4
> y <- 1 + x1 + x2 + x3 + x4 + c(-.5, .5, .5, -.5, 0, .5, -.5, -.5, .5)
```

Clearly, `x3` and `x4` depend linearly on `x1` and `x2` and the intercept. Let us fit two versions of the same model to these data, entering the predictors in different orders, and compare the regression coefficients:

```
> mod1234 <- lm(y ~ x1 + x2 + x3 + x4)
> mod4321 <- lm(y ~ x4 + x3 + x2 + x1)
> zapsmall(rbind(b1234 = coef(mod1234), b4321 = coef(mod4321)[c(1, 5:2)]))

      (Intercept) x1 x2 x3 x4
b1234           5  3  0 NA NA
b4321         -19 NA NA  3  6
```

Note that in each model, two regression coefficients are `NA`. This indicates that the associated predictors were excluded due to their linear dependence on the others.

The problem that concerns us is making predictions on new values of the predictors. Here are some predictor combinations to try:

```
> testset <- data.frame(
+              x1 = c(3,  6,  6,  0,  0,  1),
+              x2 = c(1,  2,  2,  0,  0,  2),
+              x3 = c(7, 14, 14,  0,  0,  3),
+              x4 = c(2,  4,  0,  4,  0,  4))
```

And here is what happens when we make the predictions:

```
> cbind(testset,
+     pred1234 = predict(mod1234, newdata = testset),
+     pred4321 = suppressWarnings(predict(mod4321, newdata = testset)))

  x1 x2 x3 x4 pred1234 pred4321
1  3  1  7  2       14       14
2  6  2 14  4       23       47
```

```
3  6  2 14  0       23       23
4  0  0  0  4        5        5
5  0  0  0  0        5      -19
6  1  2  3  4        8       14

Warning message:
In predict.lm(mod1234, new = testset) :
  prediction from a rank-deficient fit may be misleading
```

Note that the two models produce identical predictions in some cases (rows 1, 3, and 4), and different ones in the others. There is also a warning message (we suppressed this the second time) telling us that this could happen.

It happens that cases 1, 3, and 4 are estimable, and the others are not. It would be helpful to know which predictions to trust, rather than a vague warning. The `epredict` function in **estimability** (Lenth, 2015) accomplishes this:

```
> require("estimability")
> rbind(epred1234 = epredict(mod1234, newdata = testset),
+       epred4321 = epredict(mod4321, newdata = testset))

           1  2  3 4  5  6
epred1234 14 NA 23 5 NA NA
epred4321 14 NA 23 5 NA NA
```

The results for non-estimable cases are indicated by `NA`s. Note that in both models, the same new-data cases are identified as estimable, and they are the ones that yield the same predictions. Note also that estimability was determined separately from each model. We do not need to compare two different fits to determine estimability.

Adding estimability checking to a modeling package

It is a simple matter to add estimability checking to the `predict` method(s) in a new or existing package.

1. The package should import the **estimability** package.
2. Use `estimability::nonest.basis` to obtain the basis for non-estimable functions of the regression coefficients, preferably in the model-fitting code.
3. In the `predict` method, call `estimability::is.estble` on the model matrix for new data. (It is not necessary to check estimability of predictions on the original data.)

This is implemented in code in a manner something like this:

```
fitmodel <- function(...) {
    ...
        code to create:
            'X' (the fixed-effects model matrix),
            'QR' (QR decomp of 'X'),
            'object' (the fitted model object)
    ...
    object$nonest <- estimability::nonest.basis(QR)
    object
}

predict.mymodel <- function(object, newdata, tol = 1e-8, ...) {
    ...
    if(!missing(newdata)) {
        ...
            code to create:
                'newX' (the model matrix for 'newdata')
        ...
        estble <- estimability::is.estble(newX, object$nonest, tol)

        ...
            code to flag non-estimable predictions
        ...
    }
    ...
}
```

The `nonest.basis` function returns a matrix whose columns span the null space of the model matrix **X**. If, for some reason, the QR decomposition is not used in model fitting, there is also a `nonest.basis` method that can be called with **X** itself. But if the QR decomposition of **X** is available, it is most efficient to use it.

The `is.estble` function tests each row of its first argument for estimability against the null basis provided in its second argument, and returns a logical vector. A third argument may be added to adjust the tolerance used in this test. It is important to remember that *all columns* of the model matrix be included. In particular, do not exclude the columns corresponding to `NA`s in the coefficient estimates, as they are needed for the estimability testing. Typically, the results of `is.estble` would be used to skip any non-estimable predictions and replace them with `NA`.

If there is not a rank deficiency in the model, `nonest.basis` returns `estimability::all.estble`, which is a trivial null basis (still of 'matrix' class) with which `is.estble` will return all `TRUE`. The `nonest.basis` function, when called with a 'qr' object, can immediately detect if there is no rank deficiency and will return `all.estble`. If your model-fitting algorithm does not use the QR decomposition, it may be worth including additional code to check for non-singular models, in which case it sets the null basis to `estimability::all.estble`, rather than have `nonest.basis` perform the additional computation to figure this out.

The sample code above is typical for the S3 object model. If S4 objects are being used, you may want to instead include a slot of class 'matrix' for `nonest`; or incorporate `nonest` as part of some other slot.

To illustrate briefly, consider the previous example. The null basis for `mod1234` is obtained as follows:

```
> (nb <- nonest.basis(mod1234$qr))

            [,1]       [,2]
[1,]  0.29555933 -0.9176629
[2,]  0.76845426  0.2294157
[3,] -0.48767290 -0.2294157
[4,] -0.28078136  0.0000000
[5,] -0.07388983  0.2294157
```

and we can test estimability for the data in `testset` by converting it to a matrix and appending a column for the intercept:

```
> newX <- cbind(1, as.matrix(testset))
> is.estble(newX, nb)

[1]  TRUE FALSE  TRUE  TRUE FALSE FALSE
```

Theory of estimability

The theory of estimability in linear models is well established, and can be found in almost any text on linear models, such as classics like Searle (1997) and Seber and Lee (2003). In this discussion, I will make specific references to portions of a more recent reference, Monahan (2008).

Before delving in, though, it is worth noting (based on what is said in some contributed packages' documentation and code) that there seems to be some confusion in the R community between non-estimability of a linear function $\boldsymbol{\lambda}'\boldsymbol{\beta}$ and the placement of nonzero λ_j coefficients relative to the positions of `NA` values in the estimate **b**. It is not possible to assess estimability with this information. Note, for instance, in the example in the Introduction, we obtained two different **b**s. Between them we can find an `NA` in the estimates for every predictor except the intercept. In fact, while `NA` usually refers to unknown values, that is not the case here. An `NA` regression coefficient signals a coefficient that is constrained to zero in order to obtain an estimate. And there is nothing wrong with multiplying some of the λ_j by zero. To assess estimability of $\boldsymbol{\lambda}'\boldsymbol{\beta}$, we must look at *all* of the elements of $\boldsymbol{\lambda}$, even those corresponding to `NA`s in the estimates.

General results

When **X** is not full-rank, the solution **b** to the normal equations is not unique. However, the predicted values, **Xb**, are uniquely estimable; that is, for any two solutions $\mathbf{b}_1$ and $\mathbf{b}_2$, $\mathbf{X}\mathbf{b}_1 = \mathbf{X}\mathbf{b}_2$. Note that the

ith element of $\mathbf{Xb}$ is $\mathbf{x}_i'\mathbf{b}$ where $\mathbf{x}_i'$ is the ith row of $\mathbf{X}$. Thus, $\mathbf{x}_i'\boldsymbol{\beta}$ is estimable for each i, and so are any linear combinations of these. Indeed, $\boldsymbol{\lambda}'\boldsymbol{\beta}$ is estimable if and only if $\boldsymbol{\lambda}'$ is in the row space of $\mathbf{X}$—or equivalently, $\boldsymbol{\lambda}$ is in the column space of $\mathbf{X}'$ (Monahan, 2008, Result 3.1).

From the above result, we can establish estimability of $\boldsymbol{\lambda}'\boldsymbol{\beta}$ either by showing that $\boldsymbol{\lambda}$ is in the column space of $\mathbf{X}'$, or by showing that it is orthogonal to the null space of $\mathbf{X}$ (Monahan, 2008, Methods 3.2 and 3.3, respectively). One way to implement the second idea is to note that if $(\mathbf{X}'\mathbf{X})^-$ is any generalized inverse of $\mathbf{X}'\mathbf{X}$, then $\mathbf{P} = (\mathbf{X}'\mathbf{X})(\mathbf{X}'\mathbf{X})^-$ is a projection onto the column space of $\mathbf{X}'$, and so $\mathbf{I} - \mathbf{P}$ is a projection onto the null space of $\mathbf{X}$. The columns of $\mathbf{I} - \mathbf{P}$ thus comprise a basis for this null space. Accordingly, estimability of $\boldsymbol{\lambda}'\boldsymbol{\beta}$ can be determined by showing that $\boldsymbol{\lambda}'(\mathbf{I} - \mathbf{P}) = \mathbf{0}'$.

SAS uses $\mathbf{I} - \mathbf{P}$, as described above, to test estimability. Of course, in computation we need to set a tolerance for being close enough to $\mathbf{0}$. SAS deems $\boldsymbol{\lambda}'\boldsymbol{\beta}$ non-estimable if $\boldsymbol{\lambda} \neq \mathbf{0}$ and $\max|\boldsymbol{\lambda}'(\mathbf{I} - \mathbf{P})| > \psi \cdot \max|\boldsymbol{\lambda}|$, where ψ is a tolerance factor with a default value of 10^{-4}. For details, see SAS Institute Inc. (2013), the chapter on "Shared Concepts," documentation for the `ESTIMATE` statement and its `SINGULAR` option.

Methods used by estimability

In the **estimability** package, we obtain the null basis in a different way than shown above, in part because the QR decomposition of $\mathbf{X}$ is already (usually) available in an 'lm' object. Suppose that $\mathbf{X}$ is $n \times p$, then we can write $\mathbf{X} = \mathbf{QR}$, where $\mathbf{Q}$ ($n \times p$) has orthonormal columns, and $\mathbf{R}$ ($p \times p$) is upper triangular with nonzero diagonal elements. If $\mathbf{X}$ is of rank $r < p$, then columns are pivoted so that the linearly dependent ones come last, and we only need r columns of $\mathbf{Q}$ and the top r rows of $\mathbf{R}$, along with the pivoting information. Call these pivoted, down-sized matrices $\widetilde{\mathbf{Q}}$ and $\widetilde{\mathbf{R}}$ respectively; and let $\widetilde{\mathbf{X}}$ (still $n \times p$) denote $\mathbf{X}$ with its columns permuted according to the pivoting. We now have that $\widetilde{\mathbf{X}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}$, and $\widetilde{\mathbf{R}}$ comprises the first r rows of an upper triangular matrix. Observe that each row of $\widetilde{\mathbf{X}}$ is thus a linear combination of the rows of $\widetilde{\mathbf{R}}$—that is, the row space of $\widetilde{\mathbf{X}}$ is the same as the row space of $\widetilde{\mathbf{R}}$. So the much smaller matrix $\widetilde{\mathbf{R}}$ has everything we need to know to determine estimability.

Now, let $d = p - r$ denote the rank deficiency, and define

$$\mathbf{S} = \left[\begin{array}{c|c} \widetilde{\mathbf{R}}'_{p\times r} & \begin{array}{c} \mathbf{0}_{r\times d} \\ \mathbf{I}_{d\times d} \end{array} \end{array} \right]$$

Then $\mathbf{S}$ is an upper triangular matrix with nonzero diagonal—hence it is of full rank $p = r + d$. Let $\mathbf{S} = \mathbf{TU}$ be the QR decomposition of $\mathbf{S}$. We then have that $\mathbf{T}$ has orthonormal columns, and in fact it is a Gram-Schmidt orthonormalization of $\mathbf{S}$. Accordingly, the first r columns of $\mathbf{T}$ comprise an orthonormalization of the columns of $\widetilde{\mathbf{R}}'$, and thus they form a basis for the row space of $\widetilde{\mathbf{R}}$ and hence of $\widetilde{\mathbf{X}}$. Letting $\widetilde{\mathbf{N}}$ denote the last d columns of $\mathbf{T}$, we have that $\widetilde{\mathbf{N}}$ has orthonormal columns, all of which are orthogonal to the first r columns of $\mathbf{T}$ and hence to the row space of $\widetilde{\mathbf{R}}$ and $\widetilde{\mathbf{X}}$. It follows that $\widetilde{\mathbf{N}}$ is an orthonormal basis for the null space of $\widetilde{\mathbf{X}}$. After permuting the rows of $\widetilde{\mathbf{N}}$ according to the original pivoting in the QR decomposition of $\mathbf{X}$, we obtain a basis $\mathbf{N}$ for the null space of $\mathbf{X}$.

To test estimability of $\boldsymbol{\lambda}'\boldsymbol{\beta}$, first compute $\mathbf{z}' = \boldsymbol{\lambda}'\mathbf{N}$, which is a d-vector. Theoretically, $\boldsymbol{\lambda}'\boldsymbol{\beta}$ is estimable if and only if $\mathbf{z}'\mathbf{z} = 0$; but for computational purposes, we need to set a tolerance for its being suitably small. Again, we opt to deviate from SAS's approach, which is based on $\max|z_i|$. Instead, we deem $\boldsymbol{\lambda}'\boldsymbol{\beta}$ non-estimable when $\boldsymbol{\lambda} \neq \mathbf{0}$ and $\mathbf{z}'\mathbf{z} \geq \tau \cdot \boldsymbol{\lambda}'\boldsymbol{\lambda}$, where τ is a tolerance value. Our default value is $\tau = (10^{-4})^2 = 10^{-8}$. The rationale for this criterion is that it is rotation-invariant: We could replace $\mathbf{N} \leftarrow \mathbf{VN}$ where $\mathbf{V}$ is any $p \times p$ orthogonal matrix, and $\mathbf{z}'\mathbf{z}$ will be unchanged. Such rotation invariance seems a desirable property because the assessment of estimability does not depend on which null basis is used.

Bells and whistles

The **estimability** package's `epredict` function serves as a demonstration of adding estimability checking to the `predict` methods in the **stats** package. It works for `lm`, `aov`, `glm`, and `mlm` objects. It also provides additional options for the `type` argument of `predict`. When `newdata` is provided, `type = "estimability"` returns a logical vector showing which rows of `newdata` are estimable; and `type = "matrix"` returns the model matrix for `newdata`.

An accompanying enhancement is `eupdate`, which runs `update` on a model object and also adds a `nonest` member. Subsequent `epredict` calls on this object will use that as the null basis instead of having to reconstruct it. For example,

```
> mod123 <- eupdate(mod1234, . ~ . - x4)
> mod123$nonest

           [,1]
[1,]  0.0000000
[2,] -0.8017837
[3,]  0.5345225
[4,]  0.2672612
```

Now let's find out which predictions in `testset` are estimable with this model:

```
> epredict(mod123, newdata = testset, type = "estimability")

    1     2     3     4     5     6
 TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

Thus, more of these test cases are estimable when we drop x_4 from the model. Looking at `testset`, this makes sense because two of the previously non-estimable rows differ from estimable ones only in their values of x_4.

Conclusion

In rank-deficient linear models used for predictions on new predictor sets, it is important for users to know which predictions are to be trusted, and which of them would change if the same model had been specified in a different way. The **estimability** package provides an easy way to add this capability to new and existing model-fitting packages.

Bibliography

R. V. Lenth. *estimability: Estimability Tools for Linear Models*, 2015. URL http://CRAN.R-project.org/package=estimability. R package version 1.1. [p196]

J. F. Monahan. *A Primer on Linear Models*. Chapman & Hall/CRC, 2008. [p197, 198]

SAS Institute Inc. *SAS/STAT Software, Version 13.2*. SAS Institute Inc., Cary, NC, 2013. URL http://www.sas.com/. [p198]

S. R. Searle. *Linear Models*. Wiley Classics. Wiley-Interscience, 1997. [p197]

G. A. F. Seber and A. J. Lee. *Linear Regression Analysis*. John Wiley & Sons, Inc., second edition, 2003. [p197]

Russell V. Lenth
The University of Iowa
241 Schaeffer Hall
Iowa City, IA 52242
USA
russell-lenth@uiowa.edu

R Foundation News

by Kurt Hornik

Donations and new members

Donations

Kevin Tappe (Germany)
Hrishikesh D. Vinod (USA)

New supporting institutions

Department of Mathematical Sciences, Aalborg University, Denmark

New supporting members

Michael Blanks (USA)
Bernhard Brabec (Germany)
Jay Emerson (USA)
Matthias Haeni (Switzerland)
Jesse Krijthe (Netherlands)
Hans Mielke (Germany)
Hannes Mühleisen (Netherlands)

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Changes on CRAN

2015-01-01 to 2015-05-31

by Kurt Hornik and Achim Zeileis

New packages in CRAN task views

Bayesian **BayesSummaryStatLM**, **SamplerCompare**, **coalescentMCMC**, **matchingMarkets**.

ChemPhys **CRAC**, **CosmoPhotoz**, **RobPer**, **UPMASK**, **astrodatR**, **astrolibR**, **snapshot**, **speaq**.

Cluster **flexCWM**, **protoclust**.

DifferentialEquations **odeintr**.

Distributions **DiscreteInverseWeibull**, **DistributionUtils**, **GB2**, **GMD**, **Newdistns**, **PDQutils**, **ParetoPosStable**, **RMKdiscrete**, **SMR**, **ald**, **cmvnorm**, **condMVNorm**, **hermite**, **ihs**, **mvrtn**, **sadists**, **sgt**, **sld**, **tolerance**, **tsallisqexp**.

Econometrics **BMA**, **BMS**, **PANICr**, **Paneldata**, **Rchoice**, **brglm**, **frm**, **glmx**, **gmnl**, **ivbma**, **ivfixed**, **ivlewbel**, **ivpack**, **ivpanel**, **ivprobit**, **lavaan**, **matchingMarkets**, **multiwayvcov**, **pampe**, **panelAR**, **pdR**, **pglm**, **psidR**, **semsfa**, **wahc**, **xts**.

Finance **TAQMNGR**, **gets**, **matchingMarkets**, **restimizeapi**.

Graphics **hexbin**.

HighPerformanceComputing **Rborist**, **gmatrix**.

MedicalImaging **KATforDCEMRI***, **MRIaggr**, **brainR**, **fslr**, **neuRosim***, **neuroim***.

MetaAnalysis **CopulaREMADA**, **EasyStrata**, **MultiMeta**, **forestplot**, **metaRNASeq**.

NumericalMathematics **Brobdingnag**, **Deriv**.

OfficialStatistics **saeSim**, **simPop**, **stringdist**, **synthpop**.

Optimization **CEoptim**, **cccp**, **cec2005benchmark**, **cec2013**, **localsolver**, **matchingMarkets**.

Phylogenetics **BioGeoBEARS**, **DAMOCLES**, **bayou**, **cati**, **convevol**, **ggplot2**, **ips**, **rncl**.

Psychometrics **SNSequate**, **flexmix**, **kcirt**, **kequate**, **medflex**, **missMDA**, **nlsem**.

Spatial **vec2dtransf**, **wkb**.

TimeSeries **BNPTSclust**, **HarmonicRegression**, **RGENERATE**, **acp**, **bayesDccGarch**, **cents**, **dygraphs**, **forecTheta**, **gtop**, **hts**, **hwwntest**, **multitaper**, **sae2**, **strucchange**, **trend**, **tscount**.

WebTechnologies **blsAPI**, **chromer**, **gistr**, **urltools**.

(* = core package)

New contributed packages

ABCanalysis Computed ABC Analysis. Authors: Michael Thrun, Jorn Lotsch, Alfred Ultsch.

ADPclust Fast Clustering Using Adaptive Density Peak Detection. Authors: Yifan “Ethan” Xu, Xiao-Feng Wang.

ALTopt Optimal Experimental Designs for Accelerated Life Testing. Authors: Kangwon Seo [aut, cre], Rong Pan [aut].

ATE Inference for Average Treatment Effects using Covariate Balancing. Authors: Asad Haris and Gary Chan.

Ake Associated Kernel Estimations. Authors: W. E. Wansouwé, S. M. Somé and C. C. Kokonendji.

AnglerCreelSurveySimulation Simulate a Bus Route Creel Survey of Anglers. Author: Steven Ranney.

ApacheLogProcessor Process the Apache Web Server Log Combined Files. Author: Diogo Silveira Mendonca.

ArArRedux Rigorous Data Reduction and Error Propagation of '40Ar/39Ar' Data. Author: Pieter Vermeesch [aut, cre].

ArfimaMLM Arfima-MLM Estimation For Repeated Cross-Sectional Data. Authors: Patrick Kraft [aut, cre], Christopher Weber [ctb].

AsynchLong Regression Analysis of Sparse Asynchronous Longitudinal Data. Authors: Hongyuan Cao, Donglin Zeng, Jason P. Fine, and Shannon T. Holloway.

BACA Bubble Chart to Compare Biological Annotations by using DAVID. Authors: Vittorio Fortino and Dario Greco.

BCRA Breast Cancer Risk Assessment. Author: Fanni Zhang.

BNPTSclust A Bayesian Nonparametric Algorithm for Time Series Clustering. Authors: Martell-Juarez, D.A. & Nieto-Barajas, L.E. In view: *TimeSeries*.

BSGS Bayesian Sparse Group Selection. Authors: Kuo-Jung Lee and Ray-Bin Chen.

BayesSummaryStatLM MCMC Sampling of Bayesian Linear Models via Summary Statistics. Authors: Evgeny Savel'ev, Alexey Miroshnikov, Erin Conlon. In view: *Bayesian*.

BinOrdNonNor Concurrent Generation of Binary, Ordinal and Continuous Data. Authors: Yue Wang, Hakan Demirtas.

Binarize Binarization of One-Dimensional Data. Authors: Stefan Mundus, Christoph Müssel, Ludwig Lausser, Tamara J. Blätte, Martin Hopfensitz, Hans A. Kestler.

BinaryEPPM Mean and Variance Modeling of Binary Data. Authors: David M Smith, Malcolm J Faddy.

BivarP Estimating the Parameters of Some Bivariate Distributions. Author: Josef Brejcha.

CAM Causal Additive Model. Authors: Jonas Peters and Jan Ernest.

CARBayesdata Data Sets Used in the Vignette Accompanying the **CARBayes** Package. Author: Duncan Lee.

CARLIT Ecological Quality Ratios Calculation and Plot. Authors: Danilo Pecorino, Gina de la Fuente Mancebo, Xavier Torras.

CEGO Combinatorial Efficient Global Optimization. Author: Martin Zaefferer.

CEoptim Cross-Entropy R Package for Optimization. Authors: Tim Benham and Qibin Duan and Dirk P. Kroese and Benoit Liquet. In view: *Optimization*.

CFC Cause-Specific Framework for Competing-Risk Analysis. Authors: Mansour T.A. Sharabiani, Alireza S. Mahani.

CMplot Circle Manhattan Plot. Author: LiLin-Yin.

COMMUNAL Robust Selection of Cluster Number K. Authors: Albert Chen [aut, cre], Timothy E Sweeney [aut], Olivier Gevaert [ths].

CausalFX Methods for Estimating Causal Effects from Observational Data. Authors: Ricardo Silva [cre, aut], Robin Evans [aut].

Claddis Measuring Morphological Diversity and Evolutionary Tempo. Author: Graeme T. Lloyd.

ClueR CLUster Evaluation. Author: Pengyi Yang & Raja Jothi.

ClustMMDD Variable Selection in Clustering by Mixture Models for Discrete Data. Author: Wilson Toussile.

ColorPalette Color Palettes Generator. Author: Carl Ambroselli [aut, cre].

ConSpline Partial Linear Least-Squares Regression using Constrained Splines. Author: Mary C Meyer.

Conake Continuous Associated Kernel Estimation. Authors: W. E. Wansouwé, F. G. Libengué and C. C. Kokonendji.

Copula.Markov Estimation and Statistical Process Control Under Copula-Based Time Series Models. Authors: Takeshi Emura and Ting-Hsuan Long.

CopulaREMADA Copula Mixed Effect Models for Bivariate Meta-Analysis of Diagnostic Test Accuracy Studies. Author: Aristidis K. Nikoloulopoulos. In view: *MetaAnalysis*.

CorrMixed Estimate Correlations Between Repeatedly Measured Endpoints (E.g., Reliability) Based on Linear Mixed-Effects Models. Authors: Wim Van der Elst, Geert Molenberghs, Dieter Hilgers, & Nicole Heussen.

Coxnet Regularized Cox Model. Authors: Xiang Li, Donglin Zeng and Yuanjia Wang.

DAISIE Dynamical Assembly of Islands by Speciation, Immigration and Extinction. Authors: Rampal S. Etienne, Luis M. Valente & Albert B. Phillimore.

DEEPR Dirichlet-multinomial Evolutionary Event Profile Randomization test. Author: Mark T Merilo.

DIFtree Item Focused Trees for the Identification of Items in Differential Item Functioning. Author: Moritz Berger.

DNMF Discriminant Non-Negative Matrix Factorization. Authors: Zhilong Jia [aut, cre], Xiang Zhang [aut].

DSviaDRM Exploring Disease Similarity in Terms of Dysfunctional Regulatory Mechanisms. Author: Jing Yang.

DVHmetrics Analyze Dose-Volume Histograms and Check Constraints. Authors: Daniel Wollschlaeger [aut, cre], Heiko Karle [aut], Heinz Schmidberger [ctb].

Dark The Analysis of Dark Adaptation Data. Author: Jeremiah MF Kelly.

DiagrammeR Create Diagrams and Flowcharts Using R. Authors: Knut Sveidqvist [aut, cph] (mermaid.js), Mike Bostock [aut, cph] (d3.js), Chris Pettitt [aut, cph] (d3.js), Mike Daines [aut, cph] (viz.js), Richard Iannone [aut, cre] (R interface).

DiffCorr Analyzing and Visualizing Differential Correlation Networks in Biological Data. Authors: Atsushi Fukushima, Kozo Nishida.

EFDR Wavelet-Based Enhanced FDR for Signal Detection in Noisy Images. Authors: Andrew Zammit-Mangion [aut, cre], Hsin-Cheng Huang [aut].

ELYP Empirical Likelihood Analysis for the Cox Model and Yang-Prentice (2005) Model. Author: Mai Zhou.

ESEA Discovering the Dysregulated Pathways based on Edge Set Enrichment Analysis. Authors: Junwei Han, Xinrui Shi, Chunquan Li.

EW Edgeworth Expansion. Author: H.R.Law.

EcoGenetics Analysis of Phenotypic, Genotypic and Environmental Data. Authors: Leandro Roser, Juan Vilardi, Beatriz Saidman and Laura Ferreyra.

EcoSimR Null Model Analysis for Ecological Data. Authors: Nick Gotelli [aut], Edmund Hart [aut, cre], Aaron Ellison [aut].

EffectLiteR Average and Conditional Effects. Authors: Axel Mayer [aut, cre], Lisa Dietzfelbinger [ctb].

EntropyExplorer Tools for Exploring Differential Shannon Entropy, Differential Coefficient of Variation and Differential Expression. Authors: Kai Wang, Charles A. Phillips, Arnold M. Saxton and Michael A. Langston.

FACTscorer Scores the FACT and FACIT Family of Patient-Reported Outcome Measures. Authors: Ray Baser [aut, cre], Ayelet Greenberg [ctb].

FCMapper Fuzzy Cognitive Mapping. Authors: Shaun Turney and Michael Bachhofer.

FSInteract Fast Searches for Interactions. Authors: Hyun Jik Kim, Rajen D. Shah.

Factoshiny Perform Factorial Analysis from **FactoMineR** with a **shiny** Application. Authors: Pauline Vaissie, Astrid Monge, Francois Husson.

FastKNN Fast k-Nearest Neighbors. Author: Gaston Besanson.

FedData Functions to Automate Downloading Geospatial Data Available from Several Federated Data Sources. Authors: R. Kyle Bocinsky and Dylan Beaudette.

FinCovRegularization Covariance Matrix Estimation and Regularization for Finance. Author: Yachen Yan [aut, cre].

GAR Authorize and Request Google Analytics Data. Author: Andrew Geisler.

GAabbreviate Abbreviating Questionnaires (or Other Measures) Using Genetic Algorithms. Authors: Luca Scrucca [aut], Baljinder K. Sahdra [aut, cre].

GDAdata Datasets for the Book "Graphical Data Analysis with R". Author: Antony Unwin.

GOplot Visualization Of Functional Analysis Data. Authors: Wencke Walter, Fatima Sanchez-Cabo.

GPlab Gaussian Process Laboratory. Authors: Yves Deville, David Ginsbourger, Olivier Roustant. Contributor: Nicolas Durrande.

GWLelast Geographically Weighted Logistic Elastic Net Regression. Authors: Daisuke Yoneoka, Eiko Saito.

GaDiFPT First Passage Time Simulation for Gaussian Diffusion Processes. Authors: A. Buonocore, M.F. Carfora.

GameTheory Cooperative Game Theory. Author: Sebastian Cano-Berlanga.

GenForImp The Forward Imputation: A Sequential Distance-Based Approach for Imputing Missing Data. Authors: Nadia Solaro, Alessandro Barbiero, Giancarlo Manzi, Pier Alda Ferrari.

GlobalFit Bi-Level Optimization of Metabolic Network Models. Author: Daniel Hartleb.

Gmisc Descriptive Statistics, Transition Plots, and More. Author: Max Gordon.

HarmonicRegression Harmonic Regression to One or more Time Series. Author: Paal O. Westermark. In view: *TimeSeries*.

HelpersMG Tools for Earth Meteorological Analysis. Author: Marc Girondot.

HighDimOut Outlier Detection Algorithms for High-Dimensional Data. Author: Cheng Fan.

HistDAWass Histogram-Valued Data Analysis. Author: Antonio Irpino [aut, cre].

Holidays Holiday and Half-Day Data, for Use with the **TimeWarp** Package. Authors: Jeffery Horner, Lars Hansen, Tony Plate.

HomoPolymer Theoretical Model to Simulate Radical Polymerization. Author: Gianmarco Polotti.

IATscores Implicit Association Test Scores Using Robust Statistics. Author: Giulio Costantini.

IBDLabels Convert Between Different IBD-State Labelling Schemes. Author: Fiona Grimson.

ICAFF Imperialist Competitive Algorithm. Authors: Farimah Houshmand and Farzad Eskandari.

Interatrix Compute Chi-Square Measures with Corrections. Authors: Aurélie Siberchicot, Eléonore Hellard, Dominique Pontier, David Fouchet and Franck Sauvage.

KMDA Kernel-Based Metabolite Differential Analysis. Authors: Xiang Zhan and Debashis Ghosh.

KOGMWU Functional Summary and Meta-Analysis of Gene Expression Data. Author: Mikhail V. Matz.

Kernelheaping Kernel Density Estimation for Heaped Data. Author: Marcus Gross.

L1pack Routines for L1 Estimation. Author: Felipe Osorio.

LDAvis Interactive Visualization of Topic Models. Authors: Carson Sievert [aut, cre], Kenny Shirley [aut].

LDPD Probability of Default Calibration. Author: Denis Surzhko.

LLSR Data Analysis of Liquid-Liquid Systems. Author: Diego F Coelho.

LPTime LP Nonparametric Approach to Non-Gaussian Non-Linear Time Series Modelling. Authors: Subhadeep Mukhopadhyay, Shinjini Nandi.

LeafArea Rapid Digital Image Analysis of Leaf Area. Author: Masatoshi Katabuchi.

LocFDRPois Functions for Performing Local FDR Estimation when Null and Alternative are Poisson. Author: Kris Sankaran [aut, cre].

MCL Markov Cluster Algorithm. Author: Martin L. Jäger.

MCMC4Extremes Posterior Distribution of Extreme Value Models in R. Authors: Fernando Ferraz do Nascimento [aut, cre], Wyara Vanesa Moura e Silva [aut, ctb].

MCTM Markov Chains Transition Matrices. Author: Alessandro Bessi.

MIIVsem Two Stage Least Squares with Model Implied Instrumental Search. Authors: Zachary Fisher and Ken Bollen.

MInt Learn Direct Interaction Networks. Authors: Surojit Biswas, Meredith McDonald, Derek S. Lundberg, Jeffery L. Dangl, Vladimir Jojic.

MLmetrics Machine Learning Evaluation Metrics. Author: Yachen Yan [aut, cre].

MRIaggr Management, Display, and Processing of Medical Imaging Data. Author: Brice Ozenne. In view: *MedicalImaging*.

MaxPro Maximum Projection Designs. Authors: Shan Ba and V. Roshan Joseph.

MazamaSpatialUtils Mazama Science Spatial Data Download and Utility Functions. Authors: Jonathan Callahan [aut, cre], Will Leahy [aut], Henry Nguyen [aut].

MetNorm Statistical Methods for Normalizing Metabolomics Data. Author: Alysha M De Livera.

MiRSEA Discovering the Risk Pathways Based on microRNA. Authors: Junwei Han, Siyao Liu.

MixAll Clustering using Mixture Models. Author: Serge Iovleff [aut, cre].

MultiMeta Meta-analysis of Multivariate Genome Wide Association Studies. Author: Dragana Vuckovic. In view: *MetaAnalysis*.

MultiRR Bias, Precision, and Power for Multi-Level Random Regressions. Author: Yimen G. Araya-Ajoy.

NAM Nested Association Mapping Analysis. Authors: Alencar Xavier, William Muir, Katy Rainey, Tiago Pimenta, Qishan Wang, Shizhong Xu.

NNTbiomarker Calculate Design Parameters for Biomarker Validation Studies. Author: Roger Day.

NORRRM Geochemical Toolkit for R. Author: Renee Gonzalez Guzman.

NPC Nonparametric Combination of Hypothesis Tests. Author: Devin Caughey [aut, cre].

NSUM Network Scale Up Method. Authors: Rachael Maltiel and Aaron J. Baraff.

NormalLaplace The Normal Laplace Distribution. Authors: David Scott, Jason Shicong Fu and Simon Potter.

OpenMx Extended Structural Equation Modelling. Authors: Steven M. Boker [aut], Michael C. Neale [aut], Hermine H. Maes [aut], Michael J. Wilde [ctb], Michael Spiegel [aut], Timothy R. Brick [aut], Ryne Estabrook [aut], Timothy C. Bates [aut], Paras Mehta [ctb], Timo von Oertzen [ctb], Ross J. Gore [aut], Michael D. Hunter [aut], Daniel C. Hackett [ctb], Julian Karch [ctb], Andreas M. Brandmaier [ctb], Joshua N. Pritikin [aut, cre], Mahsa Zahery [aut], Robert M. Kirkpatrick [aut], Yang Wang [ctb].

PAFit Nonparametric Estimation of Preferential Attachment and Node Fitness in Temporal Complex Networks. Authors: Thong Pham, Paul Sheridan, Hidetoshi Shimodaira.

PDQutils PDQ Functions via Gram Charlier, Edgeworth, and Cornish Fisher Approximations. Author: Steven E. Pav [aut, cre]. In view: *Distributions*.

PGRdup Discover Probable Duplicates in Plant Genetic Resources Collections. Authors: J. Aravind [aut, cre], J. Radhamani [aut], Kalyani Srinivasan [aut], B. Ananda Subhash [aut], R. K. Tyagi [aut].

PPtreeViz Projection Pursuit Classification Tree Visualization. Author: Eun-Kyung Lee.

PhyloMeasures Fast and Exact Algorithms for Computing Phylogenetic Biodiversity Measures. Authors: Constantinos Tsirogiannis [aut, cre], Brody Sandel [aut].

PogromcyDanych PogromcyDanych / DataCrunchers is the Masive Online Open Course that Brings R and Statistics to the People. Author: Przemyslaw Biecek.

PoisBinNonNor Data Generation with Poisson, Binary and Continuous Components. Authors: Gul Inan, Hakan Demirtas.

PoisBinOrd Data Generation with Poisson, Binary and Ordinal Components. Authors: Gul Inan, Hakan Demirtas.

PoisBinOrdNonNor Generation of up to Four Different Types of Variables. Authors: Rachel Nordgren, Hakan Demirtas.

PoisBinOrdNor Data Generation with Poisson, Binary, Ordinal and Normal Components. Authors: Yiran Hu, Hakan Demirtas.

PoisNonNor Simultaneous Generation of Count and Continuous Data. Authors: Yaru Shi, Hakan Demirtas.

PopVar Genomic Breeding Tools: Genetic Variance Prediction and Cross-Validation. Authors: Tyler Tiede [aut, cre], Mohsen Mohammadi [ctb], Kevin P. Smith [ctb].

PortfolioAnalytics Portfolio Analysis, Including Numerical Methods for Optimization of Portfolios. Authors: Brian G. Peterson [cre, aut, cph], Peter Carl [aut, cph], Kris Boudt [ctb, cph], Ross Bennett [ctb, cph], Hezky Varon [ctb], Guy Yollin [ctb], R. Douglas Martin [ctb].

QCAfalsePositive Tests for Type I Error in Qualitative Comparative Analysis (QCA). Author: Bear Braumoeller.

QCAtools Helper functions for QCA in R. Author: Jirka Lewandowski [aut, cre].

RANN.L1 Fast Nearest Neighbour Search (Wraps ANN Library) Using L1 Metric. Authors: Sunil Arya and David Mount (for ANN), Samuel E. Kemp, Gregory Jefferis, Kirill Müller.

RBPcurve The Residual-based Predictiveness Curve. Authors: Giuseppe Casalicchio, Bernd Bischl.

RDML Importing Real-Time Thermo Cycler (qPCR) Data from RDML Format Files. Authors: Konstantin A. Blagodatskikh [cre, aut], Stefan Roediger [aut], Michal Burdukiewicz [aut].

RDota Data Analysis Toolbox for Dota2. Author: Xiao Lei.

RESS Integrates R and Essentia. Author: Ben Waxer.

REST RcmdrPlugin Easy Script Templates. Author: De Troyer Ewoud.

RFgroove Importance Measure and Selection for Groups of Variables with Random Forests. Author: Baptiste Gregorutti.

RFmarkerDetector Multivariate Analysis of Metabolomics Data using Random Forests. Authors: Piergiorgio Palla, Giuliano Armano.

RGENERATEPREC Tools To Generate Daily-Precipitation Time Series. Author: Emanuele Cordano.

RJafroc Analysis of Data Acquired Using the Receiver Operating Characteristic Paradigm and Its Extensions. Authors: Xuetong Zhai [aut, cre], Dev Chakraborty [aut, ths].

RLumShiny **shiny** Applications for the R Package **Luminescence**. Authors: Christoph Burow [aut, cre], Jan Odvarko [cph] (jscolor.js), AnalytixWare [cph] (ShinySky package).

ROCS Receiver Operating Characteristics Surface. Author: Tianwei Yu.

RPEnsemble Random Projection Ensemble Classification. Authors: Timothy I. Cannings and Richard J. Samworth.

RPPairwiseDesign Resolvable partially pairwise balanced design and Space-filling design via association scheme. Authors: Mohamed Laib, Imane Rezgui and Zebida Gheribi-Aoulmi.

RRTCS Randomized Response Techniques for Complex Surveys. Authors: Beatriz Cobo Rodríguez, María del Mar Rueda García, Antonio Arcos Cebrián.

RSPS RNA-Seq Power Simulation. Authors: Milan Bimali, Joseph Usset, Brooke L. Fridley.

RVFam Rare Variants Association Analyses with Family Data. Authors: Ming-Huei Chen and Qiong Yang.

RadTran Radon and Soil Gas Transport in 2D Porous Medium. Author: Francisco Lopes.

RandomFieldsUtils Utilities for the Simulation and Analysis of Random Fields. Authors: Martin Schlather [aut, cre], Reinhard Furrer [ctb].

Rborist Extensible, Parallelizable Implementation of the Random Forest Algorithm. Author: Mark Seligman. In view: *HighPerformanceComputing*.

RcellData Example Dataset for **Rcell** Package. Author: Alan Bush.

RcppAPT Rcpp Interface to the APT Package Manager. Author: Dirk Eddelbuettel.

RcppStreams Rcpp Integration of the Streamulus DSEL for Stream Processing. Author: Dirk Eddelbuettel.

RcppTOML Rcpp Bindings to Parser for Tom's Obvious Markup Language. Author: Dirk Eddelbuettel.

Rlibeemd Ensemble Empirical Mode Decomposition (EEMD) and Its Complete Variant (CEEMDAN). Authors: Jouni Helske [aut, cre] (R interface), Perttu Luukko [aut] (Original libeemd C library).

Rlinkedin Access to the LinkedIn API via R. Author: Michael Piccirilli.

Rmonkey A Survey Monkey R Client. Author: Thomas J. Leeper.

Rsampletrees Sampletrees Input/Output Processing. Authors: Kelly Burkett, Brad McNeney, Jinko Graham.

Rtts Convert Text into Speech. Author: Xiaodong Deng.

RxnSim Functions to Compute Chemical Reaction Similarity. Author: Varun Giri [aut, cre].

SAENET A Stacked Autoencoder Implementation with Interface to **neuralnet**. Authors: Stephen Hogg [aut, cre], Eugene Dubossarsky [aut].

SAMUR Stochastic Augmentation of Matched Data Using Restriction Methods. Authors: Mansour T.A. Sharabiani, Alireza S. Mahani.

SDDE Shortcuts, Detours and Dead Ends Path Types in Genome Similarity Networks. Authors: Etienne Lord, Margaux Le Cam, Eric Bapteste, Vladimir Makarenkov and Francois-Joseph Lapointe.

SETPath Spiked Eigenvalue Test for Pathway data. Author: Patrick Danaher.

SID Structural Intervention Distance. Author: Jonas Peters.

SLOPE Sorted L1 Penalized Estimation. Authors: Malgorzata Bogdan, Ewout van den Berg, Chiara Sabatti, Weijie Su, Emmanuel Candes, Evan Patterson.

SOMbrero SOM Bound to Realize Euclidean and Relational Outputs. Authors: Laura Bendhaiba [aut], Julien Boelaert [aut], Madalina Olteanu [aut], Nathalie Villa-Vialaneix [aut, cre].

SOUP Stochastic Ordering Using Permutations (and Pairwise Comparisons). Author: Federico Mattiello [aut, cre].

SPRT Wald's Sequential Probability Ratio Test. Author: Stephane Mikael Bottine.

SRCS Statistical Ranking Color Scheme for Multiple Pairwise Comparisons. Author: Pablo J. Villacorta.

SVMMatch Causal Effect Estimation and Diagnostics with Support Vector Machines. Author: Marc Ratkovic.

SWMPr Retrieving, Organizing, and Analyzing Estuary Monitoring Data. Author: Marcus W. Beck [aut, cre].

Sabermetrics Sabermetrics Functions For Baseball Analytics. Author: Peter Xenopoulos.

Scale Likert Type Questionnaire Item Analysis. Author: Nikolaos Giallousis.

SegCorr Detecting Correlated Genomic Regions. Authors: Eleni Ioanna Delatola, Emilie Lebarbier, Tristan Mary-Huard, Francois Radvanyi, Stephane Robin, Jennifer Wong.

SensusR Sensus Analytics. Author: Matthew S. Gerber.

ShapeSelectForest Shape Selection for Landsat Time Series of Forest Dynamics. Authors: Mary C. Meyer, Xiyue Liao, Elizabeth Freeman, Gretchen G. Moisen.

SimilarityMeasures Trajectory Similarity Measures. Author: Kevin Toohey.

SocialPosition Social Position Indicators Construction Toolbox. Author: Julie Falcon.

SpatPCA Regularized Principal Component Analysis for Spatial Data. Authors: Wen-Ting Wang and Hsin-Cheng Huang.

SpatialPosition Spatial Position Models. Authors: Timothee Giraud [cre, aut], Hadrien Commenges [aut], Joel Boulier [ctb].

SpecHelpers Spectroscopy Related Utilities. Authors: Bryan A. Hanson DePauw University, Greencastle Indiana USA.

StanHeaders C++ Header Files for Stan. Authors: Stan Development Team [aut, cph], Joshua N. Pritikin [cre, com].

StatMeasures Easy Data Manipulation, Data Quality and Statistical Checks. Author: Akash Jain.

SubpathwayGMir Identify Metabolic Subpathways Mediated by MicroRNAs. Authors: Li Feng, Chunquan Li and Xia Li.

SurvCorr Correlation of Bivariate Survival Times. Authors: Meinhard Ploner, Alexandra Kaider and Georg Heinze.

SurvLong Analysis of Proportional Hazards Model with Sparse Longitudinal Covariates. Authors: Hongyuan Cao, Mathew M. Churpek, Donglin Zeng, Jason P. Fine, and Shannon T. Holloway.

TAQMNGR Manage Tick-by-Tick Transaction Data. Authors: Francesco Calvori, Fabrizio Cipollini, Giampiero M. Gallo and 'gzstream' authors. In view: *Finance*.

TDAmapper Analyze High-Dimensional Data Using Discrete Morse Theory. Authors: Paul Pearson [aut, cre, trl], Daniel Muellner [aut, ctb], Gurjeet Singh [aut, ctb].

TDCor Gene Network Inference from Time-Series Transcriptomic Data. Author: Julien Lavenus.

TKF Pairwise Distance Estimation with TKF91 and TKF92 Model. Author: Ge Tan.

TRD Transmission Ratio Distortion. Author: Lam Opal Huang.

TROM Transcriptome Overlap Measure. Authors: Jingyi Jessica Li, Wei Li.

TSPred Functions for Baseline-Based Time Series Prediction. Authors: Rebecca Pontes Salles [aut, cre, cph], Eduardo Ogasawara [ths].

TSmisc **TSdbi** Extensions to Wrap Miscellaneous Data Sources. Author: Paul Gilbert.

TSsdmx **TSdbi** Extension to Connect with 'SDMX'. Author: Paul Gilbert.

TTmoment Sampling and Calculating the First and Second Moments for the Doubly Truncated Multivariate *t* Distribution. Authors: Hsiu J. Ho, Tsung-I Lin, Wan-Lun Wang, Aldo M. Garay, Victor H. Lachos, and Mauricio Castro.

Thermimage Functions for Handling Thermal Images. Authors: Glenn J. Tattersall, PhD.

ThreeArmedTrials Design and Analysis of Clinical Non-inferiority or Superiority Trials with Active and Placebo Control. Author: Tobias Muetze.

TickExec Execution Functions for Tick Data Back Test. Author: HKUST.

VCA Variance Component Analysis. Author: Andre Schuetzenmeister.

VIGoR Variational Bayesian Inference for Genome-Wide Regression. Authors: Akio Onogi and Hiroyoshi Iwata.

VarSelLCM Variable Selection for Model-Based Clustering using the Integrated Complete-Data Likelihood of a Latent Class Model. Authors: Matthieu Marbac and Mohammed Sedki.

WCE Weighted Cumulative Exposure Models. Authors: Marie-Pierre Sylvestre, Marie-Eve Beauchamp, Ryan Patrick Kyle, Michal Abrahamowicz.

WaterML Fetch and Analyze Data from WaterML or CUAHSI WaterOneFlow Web Service. Author: Jiri Kadlec [aut, cre].

WhopGenome High-Speed Processing of VCF, FASTA and Alignment Data. Authors: Ulrich Wittelsbuerger [aut, cre], Heng Li [ctb], Bob Handsaker [ctb].

WikipediaR R-Based Wikipedia Client. Authors: Avner Bar-Hen [aut, cre], Louise Baschet [ctb], Francois-Xavier Jollois [ctb], Jeremie Riou [ctb].

aSPU Adaptive Sum of Powered Score Test. Authors: Il-Youp Kwak and others.

abc.data Data Only: Tools for Approximate Bayesian Computation (ABC). Authors: Csillery Katalin [aut], Lemaire Louisiane [aut], Francois Olivier [aut], Blum Michael [aut, cre].

acc Processes Accelerometer Data. Authors: Jaejoon Song, Matthew G. Cox.

adegraphics An S4 Lattice-Based Package for the Representation of Multivariate Data. Authors: Stéphane Dray and Aurélie Siberchicot, with contributions from Jean Thioulouse. Based on earlier work by Alice Julien-Laferrière.

ald The Asymmetric Laplace Distribution. Authors: Christian E. Galarza and Victor H. Lachos. In view: *Distributions*.

alphaOutlier Obtain Alpha-Outlier Regions for Well-Known Probability Distributions. Authors: Andre Rehage, Sonja Kuhnt.

analyz Model Layer for Automatic Data Analysis via CSV File Interpretation. Author: Rodrigo Buhler.

anfis Adaptive Neuro Fuzzy Inference System in R. Authors: Cristobal Fresno, Andrea S. Llera and Elmer A. Fernandez.

apex Phylogenetic Methods for Multiple Gene Data. Authors: Thibaut Jombart [aut, cre], Zhian Namir Kamvar [aut], Klaus Schliep [aut], Rebecca Harris [aut].

asVPC Average Shifted Visual Predictive Checks. Author: Eun-Kyung Lee.

assertr Assertive Programming for R Analysis Pipelines. Author: Tony Fischetti [aut, cre].

atsd Support Querying Axibase Time-Series Database. Author: Axibase Corporation.

babar Bayesian Bacterial Growth Curve Analysis in R. Authors: Lydia Rickett, Matthew Hartley, Richard Morris and Nick Pullen.

bayesDccGarch The Bayesian Dynamic Conditional Correlation GARCH Model. Authors: Jose A Fioruci, Ricardo S Ehlers, Francisco Louzada. In view: *TimeSeries*.

betas Standardized Beta Coefficients. Author: Andrea Cantieni [aut, cre].

bigrquery An Interface to Google's BigQuery API. Authors: Hadley Wickham [aut, cre], RStudio [cph].

biogas Analyze Biogas Data and Predict Biogas Production. Authors: Sasha D. Hafner, Charlotte Rennuit, Jin Mi Triolo, and Ali Heidarzadeh Vazifehkhoran.

biorxivr Search and Download Papers from the bioRxiv Preprint Server. Author: Edmund Hart [aut, cre].

biosignalEMG Standard Processing Tools for Electromyogram Signals. Authors: J.A. Guerrero, J.E. Macias-Diaz.

blatr Send Emails Using 'Blat' for Windows. Author: Stefan Milton Bache.

blmeco Data Files and Functions Accompanying the Book "Bayesian Data Analysis in Ecology using R, BUGS and Stan". Authors: Fraenzi Korner-Nievergelt, Tobias Roth, Stefanie von Felten, Jerome Guelat, Bettina Almasi, Pius Korner-Nievergelt.

blockmodels Latent and Stochastic Block Model Estimation by a 'V-EM' Algorithm. Authors: INRA, Jean-Benoist Leger.

bootnet Bootstrap Methods for Various Network Estimation Routines. Author: Sacha Epskamp.

bootsPLS Bootstrap Subsamplings of Sparse Partial Least Squares - Discriminant Analysis for Classification and Signature Identification. Authors: Florian Rohart [aut, cre], Kim-Anh Le Cao [boss], Christine Wells [boss].

boottol Bootstrap Tolerance Levels for Credit Scoring Validation Statistics. Author: Garrett Schiltgen.

boxr Interface for the Box.com API. Author: Brendan Rocks [aut, cre].

brewdata Extracting Usable Data from the Grad Cafe Results Search. Author: Nathan Welch.

brms Bayesian Regression Models using Stan. Author: Paul-Christian Buerkner [aut, cre].

cOde Automated C Code Generation for Use with the **deSolve** and **bvpSolve** Packages. Author: Daniel Kaschek.

caretEnsemble Ensembles of Caret Models. Authors: Zachary A. Mayer [aut, cre], Jared E. Knowles [aut].

cccp Cone Constrained Convex Problems. Authors: Bernhard Pfaff [aut, cre], Lieven Vandenberghe [cph], Martin Andersen [cph], Joachim Dahl [cph]. In view: *Optimization*.

cchs Cox Model for Case-Cohort Data with Stratified Subcohort-Selection. Author: E. Jones.

cds Constrained Dual Scaling for Detecting Response Styles. Author: Pieter Schoonees [aut, cre].

cellranger Translate Spreadsheet Cell Ranges to Rows and Columns. Authors: Jennifer Bryan [cre, aut], Hadley Wickham [ctb].

cernn Covariance Estimation Regularized by Nuclear Norm Penalties. Author: Eric C. Chi.

chopthin The Chopthin Resampler. Authors: Axel Gandy and F. Din-Houn Lau.

chromer Interface to Chromosome Counts Database API. Author: Matthew Pennell [aut, cre]. In view: *WebTechnologies*.

clifro Easily Download and Visualise Climate Data from CliFlo. Author: Blake Seers [aut, cre].

climwin Climate Window Analysis. Authors: Liam D. Bailey and Martijn van de Pol.

clusterSEs Calculate Cluster-Robust p-Values and Confidence Intervals. Author: Justin Esarey [aut, cre].

clustering.sc.dp Optimal Distance-Based Clustering for Multidimensional Data with Sequential Constraint. Authors: Tibor Szkaliczki [aut, cre], J. Song [ctb].

clustertend Check the Clustering Tendency. Authors: Luo YiLan, Zeng RuTong.

coloredICA Implementation of Colored Independent Component Analysis and Spatial Colored Independent Component Analysis. Authors: Lee, S., Shen, H., Truong, Y. and Zanini, P.

commonmark Bindings to the 'CommonMark' Reference Implementation. Authors: Jeroen Ooms [aut, cre], John MacFarlane [cph].

compareC Compare Two Correlated C Indices with Right-censored Survival Outcome. Authors: Le Kang, Weijie Chen.

condMVNorm Conditional Multivariate Normal Distribution. Author: Ravi Varadhan [aut, cre]. In view: *Distributions*.

conformal Conformal Prediction for Regression and Classification. Author: Isidro Cortes.

cope Coverage Probability Excursion Sets. Author: Max Sommerfeld [aut, cre].

covr Test Coverage for Packages. Author: Jim Hester [aut, cre].

coxsei Fitting a CoxSEI Model. Author: Feng Chen.

cp4p Calibration Plot for Proteomics. Authors: Quentin Giai Gianetto, Florence Combes, Yohann Couté, Christophe Bruley, Thomas Burger.

cquad Conditional Maximum Likelihood for Quadratic Exponential Models for Binary Panel Data. Authors: Francesco Bartolucci, Claudia Pigini.

cranlogs Download Logs from the RStudio CRAN Mirror. Author: Gabor Csardi [aut, cre].

crimelinkage Statistical Methods for Crime Series Linkage. Authors: Michael Porter [aut, cre], Brian Reich [aut].

crunch Crunch.io Data Tools. Author: Neal Richardson [aut, cre].

csrplus Methods to Test Hypotheses on the Distribution of Spatial Point Processes. Author: Sarah Smith.

daff Diff, Patch and Merge for Data.frames. Authors: Paul Fitzpatrick [aut], Edwin de Jonge [aut, cre].

data.tree Hierarchical Data Structures. Author: Christoph Glur.

ddeploy Wrapper for the Duke Deploy REST API. Author: Niall McGearailt.

decisionSupport Quantitative Support of Decision Making under Uncertainty. Authors: Lutz Göhring [cre, aut], Eike Luedeling [aut].

decode Differential Co-Expression and Differential Expression Analysis. Author: Thomas Lui [aut, cre].

denovolyzeR Statistical Analyses of De Novo Genetic Variants. Authors: James Ware [aut, cre], Jason Homsy [ctb], Kaitlin Samocha [ctb].

dga Capture-Recapture Estimation using Bayesian Model Averaging. Authors: James Johndrow, Kristian Lum, and Patrick Ball.

diagonals Block Diagonal Extraction or Replacement. Author: Bastiaan Quast [aut, cre].

discSurv Discrete Time Survival Analysis. Authors: Thomas Welchowski and Matthias Schmid.

disparityfilter Disparity Filter Algorithm of Weighted Network. Author: Alessandro Bessi.

divo Tools for Analysis of Diversity and Similarity in Biological Systems. Authors: Maciej Pietrzak, Michal Seweryn, Grzegorz Rempala.

dmm Dyadic Mixed Model for Pedigree Data. Author: Neville Jackson.

dplRCon Concordance for Dendroclimatology. Author: Maryann Pirie.

dpmr Data Package Manager for R. Author: Christopher Gandrud [aut, cre].

drat Drat R Archive Template. Author: Dirk Eddelbuettel.

dropR Analyze Drop Out of an Experiment or Survey. Authors: Matthias Bannert [aut, cre], Ulf-Dietrich Reips [aut].

dst Using Dempster-Shafer Theory. Author: Claude Boivin.

dynRB Dynamic Range Boxes. Authors: Manuela Schreyer and Wolfgang Trutschnig.

ebGenotyping Genotyping using Next Generation Sequencing Data. Authors: Na You and Gongyi Huang.

elastic General Purpose Interface to Elasticsearch. Author: Scott Chamberlain [aut, cre].

emIRT EM Algorithms for Estimating Item Response Theory Models. Authors: Kosuke Imai, James Lo, Jonathan Olmsted.

emov Eye Movement Analysis Package for Fixation and Saccade Detection. Author: Simon Schwab.

episensr Basic Sensitivity Analysis of Epidemiological Results. Author: Denis Haine [aut, cre].

esaBcv Estimate Number of Latent Factors and Factor Matrix for Factor Analysis. Authors: Art B. Owen [aut], Jingshu Wang [aut, cre].

estimability Estimability Tools for Linear Models. Author: Russell V. Lenth.

eurostat Tools for Eurostat Open Data. Authors: Lahti Leo [aut, cre], Biecek Przemyslaw [aut], Kainu Markus [aut], Huovari Janne [aut].

evolqg Tools for Evolutionary Quantitative Genetics. Authors: Ana Paula Assis, Diogo Melo, Edgar Zanella, Fabio Machado, Guilherme Garcia.

fastpseudo Fast Pseudo Observations. Authors: Dayne Batten [aut, cre], Maja Pohar Perme [ctb], Mette Gerster [ctb].

fbroc Fast Algorithms to Bootstrap Receiver Operating Characteristics Curves. Author: Erik Peter [aut, cre].

fdrDiscreteNull False Discovery Rate Procedure Under Discrete Null Distributions. Authors: Xiongzhi Chen and R.W. Doerge.

fermicatsR Fermi Large Area Telescope Catalogs. Author: Pablo Saz Parkinson [aut, cre].

fgpt Floating Grid Permutation Technique. Author: Reinder Radersma & Ben Sheldon.

fheatmap Draw Heatmaps with Colored Dendogram. Authors: Vaishali Tumulu and Sivasish Sindiri.

fitbitScraper Scrapes Data from www.fitbit.com. Author: Cory Nissen [aut, cre].

flower Tools for characterizing flowering traits. Author: Xie Wang.

forecTheta Forecasting Time Series by Theta Method. Authors: Jose Augusto Fioruci, Francisco Louzada and Bao Yiqi. In view: *TimeSeries*.

forestFloor Visualizes Random Forests with Feature Contributions. Author: Soeren Havelund Welling.

fpCompare Reliable Comparison of Floating Point Numbers. Authors: Alex M Chubaty [aut, cre], Her Majesty the Queen in Right of Canada, as represented by the Minister of Natural Resources Canada [cph].

frmpd Regression Analysis of Panel Fractional Responses. Author: Joaquim J.S. Ramalho.

fsia Import and Analysis of OMR Data from FormScanner. Author: Michela Battauz.

funtimes Functions for Time Series Analysis. Authors: Vyacheslav Lyubchich, Yulia R. Gel and Xingyu Wang.

gapminder Gapminder Data. Author: Jennifer Bryan [aut, cre].

gaselect Genetic Algorithm for Variable Selection from High-Dimensional Data. Author: David Kepplinger.

gazepath Gazepath Transforms Eye-Tracking Data into Fixations and Saccades. Author: Daan van Renswoude.

gbm2sas Convert GBM Object Trees to SAS Code. Author: John R. Dixon.

gdm Functions for Generalized Dissimilarity Modeling. Authors: Glenn Manion, Matthew Lisk, Simon Ferrier, Diego Nieto-Lugilde, Matthew C. Fitzpatrick.

gelnet Generalized Elastic Nets. Author: Artem Sokolov.

genderizeR Gender Prediction Based on First Names. Author: Kamil Wais [aut, cre].

geojsonio Convert Data from and to 'geoJSON' or 'topoJSON'. Authors: Scott Chamberlain [aut, cre], Andy Teucher [aut].

ggExtra Add Marginal Histograms to **ggplot2**, and More **ggplot2** Enhancements. Author: Dean Attali [aut, cre].

ggenealogy Visualization Tools for Genealogical Data. Authors: Lindsay Rutter, Susan Vanderplas, Di Cook.

gistr Work with GitHub Gists. Authors: Ramnath Vaidyanathan [aut], Karthik Ram [aut], Scott Chamberlain [aut, cre]. In view: *WebTechnologies*.

git2r Provides Access to Git Repositories.

glba General Linear Ballistic Accumulator Models. Author: Ingmar Visser.

glmgraph Graph-Constrained Regularization for Sparse Generalized Linear Models. Authors: Li Chen, Jun Chen.

glmm Generalized Linear Mixed Models via Monte Carlo Likelihood Approximation. Author: Christina Knudson.

gmnl Multinomial Logit Models with Random Parameters. Authors: Mauricio Sarrias [aut, cre], Ricardo Daziano [aut], Yves Croissant [ctb]. In view: *Econometrics*.

graphicalVAR Graphical VAR for Experience Sampling Data. Author: Sacha Epskamp.

graphscan Cluster Detection with Hypothesis Free Scan Statistic. Authors: Robin Loche, Benoit Giron, David Abrial, Lionel Cucala, Myriam Charras-Garrido, Jocelyn De-Goer.

greyzoneSurv Fit a Grey-Zone Model with Survival Data. Authors: Pingping Qu and John Crowley.

gromovlab Gromov-Hausdorff Type Distances for Labeled Metric Spaces. Author: Volkmar Liebscher.

groupRemMap Regularized Multivariate Regression for Identifying Master Predictors Using the GroupRemMap Penalty. Authors: Xianlong Wang, Li Qin, Hexin Zhang, Yuzheng Zhang, Li Hsu, Pei Wang.

gsheet Download Google Sheets Using Just the URL. Author: Max Conway [aut, cre].

gsw Gibbs Sea Water Functions. Authors: Dan Kelley [aut, cre, cph], Clark Richards [aut, cph], WG127 SCOR/IAPSO [aut, cph].

gtop Game-Theoretically OPtimal Reconciliation Method. Authors: Jairo Cugliari, Tim van Erven. In view: *TimeSeries*.

gvc Global Value Chains Tools. Authors: Bastiaan Quast [aut, cre], Victor Kummritz [aut].

h5 Interface to the 'HDF5' Library. Author: Mario Annau [aut, cre].

haplotypes Haplotype Inference and Statistical Analysis of Genetic Variation. Author: Caner Aktas.

haven Import SPSS, Stata and SAS Files. Authors: Hadley Wickham [aut, cre], Evan Miller [aut, cph], RStudio [cph].

hbm Hierarchical Block Matrix Analysis. Author: Yoli Shavit.

hgm Holonomic Gradient Method and Gradient Descent. Authors: Nobuki Takayama, Tamio Koyama, Tomonari Sei, Hiromasa Nakayama, Kenta Nishiyama.

hierDiversity Hierarchical Multiplicative Partitioning of Complex Phenotypes. Authors: Zachary Marion, James Fordyce, and Benjamin Fitzpatrick.

hiertest Convex Hierarchical Testing of Interactions. Authors: Jacob Bien, Noah Simon, and Rob Tibshirani.

hisse Hidden State Speciation and Extinction. Authors: Jeremy M. Beaulieu, Brian O'Meara.

histmdl A Most Informative Histogram-Like Model. Author: Jouke Witteveen.

hornpa Horn's (1965) Test to Determine the Number of Components/Factors. Author: Francis Huang.

hsdar Manage, Analyse and Simulate Hyperspectral Data in R. Authors: Lukas W. Lehnert [cre, aut], Hanna Meyer [ctb], Joerg Bendix [ctb].

httk High-Throughput Toxicokinetics. Authors: John Wambaugh and Robert Pearce, Schmitt method implementation by Jimena Davis, dynamic model adapted from code by R. Woodrow Setzer, Rabbit parameters from Nisha Sipes.

hwwntest Tests of White Noise using Wavelets. Authors: Delyan Savchev [aut], Guy Nason [aut, cre]. In view: *TimeSeries*.

icenReg Regression Models for Interval Censored Data. Author: Clifford Anderson-Bergman.

ifctools Italian Fiscal Code ('Codice Fiscale') Utilities. Author: Luca Braglia [aut, cre].

ig.vancouver.2014.topcolour Instagram 2014 Vancouver Top Colour Dataset. Author: Roland Tanglao [aut, cre].

ihs Inverse Hyperbolic Sine Distribution. Author: Carter Davis. In view: *Distributions*.

import An Import Mechanism for R. Author: Stefan Milton Bache.

inferference Methods for Causal Inference with Interference. Author: Bradley Saul.

infra An Infrastructure Proxy Function. Author: Steve Pickering.

infuser A Very Basic Templating Engine. Author: Bart Smeets.

install.load Check, Install and Load CRAN & USGS GRAN Packages. Author: maloneypatr at Stack Overflow, Irucka Embry.

interferenceCI Exact Confidence Intervals in the Presence of Interference. Author: Joseph Rigdon.

internetarchive An API Client for the Internet Archive. Author: Lincoln Mullen [aut, cre].

itsadug Interpreting Time Series and Autocorrelated Data Using GAMMs. Authors: Jacolien van Rij [aut, cre], Martijn Wieling [aut], R. Harald Baayen [aut], Hedderik van Rijn [aut].

ivmodel Statistical Inference and Diagnostics for Instrumental Variables Model. Authors: Yang Jiang, Hyunseung Kang, and Dylan Small.

ivpanel Instrumental Panel Data Models. Author: Zaghdoudi Taha. In view: *Econometrics*.

jagsUI A Wrapper Around **rjags** to Streamline JAGS Analyses. Author: Ken Kellner.

james.analysis Analysis Tools for the 'JAMES' Framework. Author: Herman De Beukelaer.

jiebaRD Chinese Text Segmentation Data for **jiebaR** Package. Author: Qin Wenfeng.

jmetrik Tools for Interacting with 'jMetrik'. Author: J. Patrick Meyer.

joint.Cox Penalized Likelihood Estimation under the Joint Cox Models Between TTP and OS for Meta-Analysis. Author: Takeshi Emura.

js Tools for Working with JavaScript in R. Author: Jeroen Ooms.

kfigr Integrated Code Chunk Anchoring and Referencing for R Markdown Documents. Author: Michael Koohafkan.

kissmig a Keep It Simple Species Migration Model. Authors: Michael P. Nobis [cre, aut], Signe Normand [ctb].

kmodR K-Means with Simultaneous Outlier Detection. Author: David Charles Howe [aut, cre].

laketemps Lake Temperatures Collected by Situ and Satellite Methods from 1985-2009. Author: Jordan S Read.

lamW Lambert-W Function. Author: Avraham Adler [aut, cph, cre].

lawn R Client for 'Turfjs' for Geospatial Analysis. Authors: Scott Chamberlain [aut, cre], Jeff Hollister [aut].

lba Latent Budget Analysis for Compositional Data. Author: Enio G. Jelihovschi Ivan Bezerra Allaman.

lbfgsb3 Limited Memory BFGS Minimizer with Bounds on Parameters. Authors: John C Nash [aut, cre], Ciyou Zhu [aut], Richard Byrd [aut], Jorge Nocedal [aut], Jose Luis Morales [aut].

learnstats An Interactive Environment for Learning Statistics. Author: Daniel Walter [aut, cre].

lfactors Factors with Levels. Author: Paul Bailey [aut, cre].

lfl Linguistic Fuzzy Logic. Author: Michal Burda.

linbin Binning and Plotting of Linearly Referenced Data. Authors: Ethan Z. Welty [aut, cre], Christian E. Torgersen [ctb], Samuel J. Brenkman [ctb], Jeffrey J. Duda [ctb], Jonathan B. Armstrong [ctb].

linkR 3D Lever and Linkage Mechanism Modeling. Author: Aaron Olsen.

listenv Environments Behaving (Almost) as Lists. Author: Henrik Bengtsson [aut, cre, cph].

lmfor Functions for Forest Biometrics. Author: Lauri Mehtatalo.

lodGWAS Genome-Wide Association Analysis of a Biomarker Accounting for Limit of Detection. Authors: Ahmad Vaez, Ilja M. Nolte, Peter J. van der Most.

loopr Uses an Archive to Amend Previous Stages of a Pipe using Current Output. Author: Brandon Taylor.

lsbclust Least-Squares Bilinear Clustering for Three-Way Data. Authors: Pieter Schoonees [aut, cre], Patrick Groenen [ctb].

manifestoR Access and Process Data and Documents of the Manifesto Project. Authors: Jirka Lewandowski [aut, cre], Nicolas Merz [aut], Sven Regel [ctb], Pola Lehmann [ctb].

mapDK Maps of Denmark. Author: Sebastian Barfort.

mapfit A Tool for PH/MAP Parameter Estimation. Author: Hiroyuki Okamura.

marl Multivariate Analysis Based on Relative Likelihoods. Author: Milan Bimali.

matchingR Gale-Shapley Algorithm in R and C++. Author: Jan Tilly.

medicalrisk Medical Risk and Comorbidity Tools for ICD-9-CM Data. Authors: Patrick McCormick [aut, cre], Thomas Joseph [aut].

metagear Comprehensive Research Synthesis Tools for Systematic Reviews and Meta-Analysis. Author: Marc J. Lajeunesse [aut, cre].

mglmn Model Averaging for Multivariate GLM with Null Models. Authors: Masatoshi Katabuchi and Akihiro Nakamura.

minimist Parse Argument Options. Authors: Jeroen Ooms, James Halliday.

mitml Tools for Multiple Imputation in Multilevel Modeling. Authors: Simon Grund [aut, cre], Alexander Robitzsch [aut], Oliver Luedtke [aut].

mixedMem Tools for Discrete Multivariate Mixed Membership Models. Authors: Y. Samuel Wang [aut, cre], Elena A. Erosheva [aut].

mixlm Mixed Model ANOVA and Statistics for Education. Authors: Kristian Hovde Liland [aut, cre], Solve Sæbø [ctb], R-Core [ctb].

mixtNB DE Analysis of RNA-Seq Data by Mixtures of NB. Authors: Elisabetta Bonafede, Cinzia Viroli.

mldr Exploratory Data Analysis and Manipulation of Multi-Label Data Sets. Authors: David Charte [cre], Francisco Charte [aut].

mlxR Simulation of Longitudinal Data. Authors: Marc Lavielle [aut, cre], Raphael Kuate [ctb], Romain Francois [ctb], Fazia Bellal [ctb].

mma Multiple Mediation Analysis. Author: Qingzhao Yu.

mmpp Various Similarity and Distance Metrics for Marked Point Processes. Authors: Hideitsu Hino, Ken Takano, Yuki Yoshikawa, and Noboru Murata.

modMax Community Structure Detection via Modularity Maximization. Authors: Maria Schelling, Cang Hui.

momentchi2 Moment-Matching Methods for Weighted Sums of Chi-Squared Random Variables. Author: Dean Bodenham.

mongolite Fast and Simple 'MongoDB' Client for R. Authors: Jeroen Ooms [aut, cre], MongoDB, Inc [cph].

mountainplot Mountain Plots, Folded Empirical Cumulative Distribution Plots. Author: Kevin Wright.

mousetrack Mouse-Tracking Measures from Trajectory Data. Authors: Moreno I. Coco and Nicholas D. Duran with contributions of Rick Dale, Denis O'Hora and Michael J. Spivey.

muir Exploring Data with Tree Data Structures. Author: Justin Alford [aut, cre].

multiAssetOptions Finite Difference Method for Multi-Asset Option Valuation. Authors: Michael Eichenberger and Carlo Rosa.

multimark Capture-Mark-Recapture Analysis using Multiple Non-Invasive Marks. Authors: Brett T. McClintock [aut, cre], Acho Arnold [ctb, cph], Barry Brown [ctb], James Lovato [ctb], John Burkardt [ctb], Cleve Moler [ctb].

multirich Calculate Multivariate Richness via UTC and sUTC. Author: Alexander Keyel.

multiway Component Models for Multi-Way Data. Author: Nathaniel E. Helwig.

mvmesh Multivariate Meshes and Histograms in Arbitrary Dimensions. Author: John P. Nolan.

mvnpermute Generate New Multivariate Normal Samples from Permutations. Author: Mark Abney.

mztwinreg Regression Models for Monozygotic Twin Data. Author: Aldo Cordova-Palomera.

nFCA Numerical Formal Concept Analysis for Systematic Clustering. Authors: Junheng Ma, Jiayang Sun, and Guo-Qiang Zhang.

nLTT Calculate the NLTT Statistic. Author: Thijs Janzen.

ncappc NCA Calculation and Population PK Model Diagnosis. Authors: Chayan Acharya [aut, cre], Andrew C. Hooker [aut], Siv Jonsson [aut], Mats O. Karlsson [aut].

neotoma Access to the Neotoma Paleoecological Database Through R. Authors: Simon J. Goring [aut, cre], Gavin L. Simpson [aut], Jeremiah P. Marsicek [ctb], Karthik Ram [aut], Luke Sosalla [ctb].

nestedRanksTest Mann-Whitney-Wilcoxon Test for Nested Ranks. Author: Douglas G. Scofield [aut, cre].

netassoc Inference of Species Associations from Co-Occurrence Data. Authors: Benjamin Blonder, Naia Morueta-Holme.

netgen Network Generator for Combinatorial Graph Problems. Author: Jakob Bossek [aut, cre].

ngramrr A Simple General Purpose N-Gram Tokenizer. Author: Chung-hong Chan.

nnlasso Non-Negative Lasso and Elastic Net Penalized Generalized Linear Models. Authors: B N Mandal and Jun Ma.

novelist NOVEL Integration of the Sample and Thresholded Correlation and Covariance Estimators. Authors: Na Huang and Piotr Fryzlewicz.

npIntFactRep Nonparametric Interaction Tests for Factorial Designs with Repeated Measures. Author: Jos Feys.

odeintr C++ ODE Solvers Compiled on-Demand. Author: Timothy H. Keitt. In view: *DifferentialEquations*.

onls Orthogonal Nonlinear Least-Squares Regression. Author: Andrej-Nikolai Spiess.

optifunset Set Options if Unset. Author: Nicholas Hamilton.

ordinalCont Ordinal Regression Analysis for Continuous Scales. Authors: Maurizio Manuguerra [aut, cre], Gillian Heller [aut].

pRF Permutation Significance for Random Forests. Author: Ankur Chakravarthy.

pacman Package Management Tool. Authors: Tyler Rinker [aut, cre, ctb], Dason Kurkiewicz [aut, ctb].

pairsD3 D3 Scatterplot Matrices. Author: Garth Tarr [aut, cre].

pampe Implementation of the Panel Data Approach Method for Program Evaluation. Author: Ainhoa Vega-Bayo. In view: *Econometrics*.

parallelSVM A Parallel-Voting Version of the Support-Vector-Machine Algorithm. Author: Wannes Rosiers.

partialAR Partial Autoregression. Author: Matthew Clegg [aut, cre, cph].

partools Tools for the **parallel** Package. Author: Norm Matloff.

patchSynctex Communication Between Editor and Viewer for Literate Programs. Authors: Jan Gleixner [aut], Daniel Hicks [ctb], Emmanuel Charpentier [aut, cre].

pathological Path Manipulation Utilities. Authors: Richard Cotton [aut, cre], Janko Thyson [ctb].

pcaBootPlot Create 2D Principal Component Plots with Bootstrapping. Author: Joshua Starmer.

perspectev Permutation of Species During Turnover Events. Authors: Kenneth B. Hoehn [aut, cre], Glen A. Sargeant [ctb].

phylocurve Phylogenetic Comparative Methods for Function-Valued and Other High-Dimensional Traits. Author: Eric W. Goolsby.

physiology Calculate Physiological Characteristics of Adults and Children. Author: Jack O. Wasey [aut, cre].

phytotools Phytoplankton Production Tools. Authors: Greg M. Silsbe, Sairah Y. Malkin.

pipe.design Dual-Agent Dose Escalation for Phase I Trials using the PIPE Design. Author: Michael Sweeting.

plantecophys Modelling and Analysis of Leaf Gas Exchange Data. Author: Remko Duursma.

plaqr Partially Linear Additive Quantile Regression. Author: Adam Maidman [cre, aut].

pogit Bayesian Variable Selection for a Poisson-Logistic Model. Authors: Michaela Dvorzak [aut, cre], Helga Wagner [aut].

pointRes Analyzing Pointer Years and Components of Resilience. Authors: Marieke van der Maaten-Theunissen and Ernst van der Maaten.

poisDoubleSamp Confidence Intervals with Poisson Double Sampling. Authors: David Kahle [aut, cre], Phil Young [aut], Dean Young [aut].

ppiPre Predict Protein-Protein Interactions Based on Functional and Topological Similarities. Authors: Yue Deng, Rongjie Shao, Gang Wang and Yuanjun Sun.

prais Prais-Winsten Estimation Procedure for AR(1) Serial Correlation. Author: Franz Mohr.

precintcon Precipitation Intensity, Concentration and Anomaly Analysis. Author: Lucas Venezian Povoa.

presens R Interface for PreSens Fiber Optic Data. Author: Matthew A. Birk.

probFDA Probabilistic Fisher Discriminant Analysis. Author: Charles Bouveyron & Camille Brunet.

probemod Statistical Tools for Probing Moderation Effects. Author: Jiat Chow Tan [aut, cre].

progress Terminal Progress Bars. Author: Gabor Csardi [aut, cre].

provenance Statistical Toolbox for Sedimentary Provenance Analysis. Author: Pieter Vermeesch [aut, cre].

pssm Piecewise Exponential Model for Time to Progression and Time from Progression to Death. Author: David A. Schoenfeld [aut, cre].

pullword R Interface to Pullword Service. Author: Tong He.

pvrank Rank Correlations. Authors: Amerise I. L., Marozzi M., Tarsitano A.

qclust Robust Estimation of Gaussian Mixture Models. Authors: Yichen Qin, Carey E. Priebe.

qrLMM Quantile Regression for Linear Mixed-Effects Models. Authors: Christian E. Galarza and Victor H. Lachos.

qrNLMM Quantile Regression for Nonlinear Mixed-Effects Models. Authors: Christian E. Galarza and Victor H. Lachos.

qrmtools Tools for Quantitative Risk Management. Authors: Marius Hofert [aut, cre], Kurt Hornik [aut].

qrng (Randomized) Quasi-Random Number Generators. Authors: Marius Hofert [aut, cre], Christiane Lemieux [aut].

qte Quantile Treatment Effects. Author: Brantly Callaway.

quanteda Quantitative Analysis of Textual Data. Authors: Kenneth Benoit [aut, cre], Paul Nulty [aut], Pablo Barberá [ctb], Kohei Watanabe [ctb], Benjamin Lauderdale [ctb].

quantification Quantification of Qualitative Survey Data. Author: Joachim Zuckarelli.

quickpsy Fits Psychometric Functions for Multiple Groups. Authors: Daniel Linares [aut, cre], Joan López-Moliner [aut].

qwraps2 Quick Wraps 2. Author: Peter DeWitt [aut, cre].

rLTP R interface to LTP-Cloud service. Author: Tong He.

rLiDAR LiDAR Data Processing and Visualization. Authors: Carlos A. Silva, Nicholas L. Crookston, Andrew T. Hudak, Lee A. Vierling.

rNMF Robust Nonnegative Matrix Factorization. Authors: Yifan Ethan Xu, Jiayang Sun.

rTableICC Random Generation of Contingency Tables. Author: Haydar Demirhan.

rUnemploymentData Data and Functions for USA State and County Unemployment Data. Author: Ari Lamstein [cre].

radiant Business Analytics using R and **shiny**. Author: Vincent Nijs [aut, cre].

radir Inverse-Regression Estimation of Radioactive Doses. Authors: David Moriña, Manuel Higueras and Pedro Puig.

ramify Additional Matrix Functionality. Author: Brandon Greenwell [aut, cre].

randNames Provide Access to Fake User Data. Author: Karthik Ram [aut, cre].

randomizr Easy to Use Tools for Common Forms of Random Assignment. Author: Alexander Coppock [aut, cre].

rankdist Distance Based Ranking Models. Author: Zhaozhi Qian.

rchallenge A Simple Datascience Challenge System. Authors: Adrien Todeschini [aut, cre], Robin Genuer [ctb].

rcorpora A Collection of Small Text Corpora of Interesting Data. Authors: Darius Kazemi, Matthew Rothenberg, Karl Swedberg, Matthew Hokanson, Nathan Lachenmyer, Aaron Marriner, Mark Sample, Casey Kolderup, Nathaniel Mitchell, Daniel D. Beck, Mike Nowak, Ryan Freebern, Ross Barclay, Ross Binden, Justin Alford, Cole Willsea, Andrew Gorman, Javier Arce, Patrick Rodriguez, Liam Cooke, Will Hankinson, K. Adam White, Garrett Miller, Zac Moody, Jordan Killpack, Brian Jones, Greg Borenstein, Noah Swartz, Nathan Black, Russell Horton, Mark Wunsch, Kay Belardinelli, Colin Mitchell, Michael Dewberry, Joe Mahoney.

rdrop2 Programmatic Interface to the Dropbox API. Author: Karthik Ram [aut, cre].

reGenotyper Detecting Mislabeled Samples in Genetic Data. Author: Yang Li.

readGenalex Read, Write, Manipulate and Convert GenAlEx-Format Genotype Files. Author: Douglas G. Scofield [aut, cre].

readr Read Tabular Data. Authors: Hadley Wickham [aut, cre], Romain Francois [aut], R Core Team [ctb], RStudio [cph].

readstata13 Import Stata 13 and 14 Data Files. Authors: Jan Marvin Garbuszus [aut], Sebastian Jeworutzki [aut, cre], R Core Team [cph].

readxl Read Excel Files. Authors: Hadley Wickham [aut, cre], RStudio [cph], Marcin Kalicinski [ctb, cph], Komarov Valery [ctb, cph], Christophe Leitienne [ctb, cph], Bob Colbert [ctb, cph], David Hoerl [ctb, cph].

rebus Build Regular Expressions in a Human Readable Way. Author: Richard Cotton [aut, cre].

redist Markov Chain Monte Carlo Methods for Redistricting Simulation. Authors: Ben Fifield, Alexander Tarr, Michael Higgins, and Kosuke Imai.

replicatedpp2w Two-Way ANOVA-Like Method to Analyze Replicated Point Patterns. Author: Marcelino de la Cruz Rot.

reproducer Reproduce Statistical Analyses and Meta-Analyses. Authors: Lech Madeyski [cre, aut], Marian Jureczko [ctb], Barbara Kitchenham [ctb].

rerddap General Purpose Client for 'ERDDAP' Servers. Author: Scott Chamberlain [aut, cre].

restimizeapi Functions for Working with the 'www.estimize.com' Web Services. Author: Thomas P. Fuller. In view: *Finance*.

retrosheet Import Professional Baseball Data from 'Retrosheet'. Author: Richard Scriven [aut, cre].

reval Repeated Function Evaluation for Sensitivity Analysis. Author: Michael C Koohafkan [aut, cre].

rglobi R Interface to Global Biotic Interactions. Authors: Jorrit Poelen [aut, cre], Stephen Gosnell [aut], Sergey Slyusarev [aut].

rgrass7 Interface Between GRASS 7 Geographical Information System and R. Authors: Roger Bivand [cre, aut], Rainer Krug [ctb], Markus Neteler [ctb].

riceware A Diceware Passphrase Implementation. Authors: Francois Michonneau [aut, cre], Arnold G. Reinhold [cph].

rivr Steady and Unsteady Open-Channel Flow Computation. Author: Michael C Koohafkan [aut, cre].

rjade A Clean, Whitespace-Sensitive Template Language for Writing HTML. Authors: Jeroen Ooms, Forbes Lindesay.

rkafka Using Apache 'Kafka' Messaging Queue Through R. Author: Shruti Gupta[aut,cre].

rkafkajars External Jars Required for Package **rkafka**. Authors: Shruti Gupta [aut, cre], Coda Hale and Yammer Inc. [ctb, cph], Sun Microsystems Inc. [ctb, cph], Marc Prud'hommeaux [ctb, cph], Paul Holser [ctb], Junit [ctb, cph], The Apache Software Foundation [ctb, cph], Stefan Groschupf [ctb, cph], Taro L.Saito [ctb], EPFL Typesafe Inc. [ctb, cph], QOS.ch [ctb, cph].

rlm Robust Fitting of Linear Model. Author: Oleg Yegorov.

rotationForest Fit and Deploy Rotation Forest Models. Authors: Michel Ballings and Dirk Van den Poel.

roughrf Roughened Random Forests for Binary Classification. Author: Kuangnan Xiong.

rprime Functions for Working with 'Eprime' Text Files. Author: Tristan Mahr.

rr Statistical Methods for the Randomized Response Technique. Authors: Graeme Blair, Yang-Yang Zhou, Kosuke Imai.

rsatscan Tools, Classes, and Methods for Interfacing with SaTScan Stand-Alone Software. Author: Ken Kleinman [aut, cre].

rscala Bi-Directional Interface Between R and Scala with Callbacks. Authors: David B. Dahl [aut, cre], Scala developers [ctb].

rtable Tabular Reporting Functions. Author: David Gohel [aut, cre].

rversions Query R Versions, Including 'r-release' and 'r-oldrel'. Authors: Gabor Csardi [aut, cre], Jeroen Ooms [ctb].

sadists Some Additional Distributions. Author: Steven E. Pav [aut, cre]. In view: *Distributions*.

sae2 Small Area Estimation: Time-series Models. Authors: Robert E. Fay, Mamadou Diallo. In view: *TimeSeries*.

saturnin Spanning Trees Used for Network Inference. Author: Loïc Schwaller.

sdwd Sparse Distance Weighted Discrimination. Authors: Boxiang Wang, Hui Zou.

searchable Tools for Custom Searches / Subsets / Slices of Named R Objects. Author: DecisionPatterns [aut, cre].

seismicRoll Fast Rolling Functions for Seismology using **Rcpp**. Authors: Jonathan Callahan [aut, cre], Rob Casey [aut], Mary Templeton [aut].

selfea Select Features Reliably with Cohen's Effect Sizes. Authors: Lang Ho Lee, Arnold Saxton, Nathan Verberkmoes.

semsfa Semiparametric Estimation of Stochastic Frontier Models. Authors: Giancarlo Ferrara and Francesco Vidoli. In view: *Econometrics*.

seroincidence Estimating Infection Rates from Serological Data. Authors: Peter Teunis [aut], Daniel Lewandowski [com, ctb], Chantal Quinten [ctb, cre].

sgPLS Sparse Group Partial Least Square Methods. Authors: Benoit Liquet and Pierre Lafaye de Micheaux.

sgd Stochastic Gradient Descent for Scalable Estimation. Authors: Dustin Tran [aut, cre], Tian Lian [aut], Panos Toulis [aut], Ye Kuang [ctb], Edoardo Airoldi [ctb].

sgt Skewed Generalized *T* Distribution. Author: Carter Davis. In view: *Distributions*.

shapeR Collection and Analysis of Otolith Shape Data. Authors: Lisa Anne Libungan [aut, cre], Snaebjorn Palsson [aut, ths].

shinyTree jsTree Bindings for **shiny**. Authors: Trestle Technology, LLC [aut], Jeff Allen [cre], Institut de Radioprotection et de Sûreté Nucléaire [cph], Ivan Bozhanov [ctb, cph], The Dojo Foundation [ctb, cph], jQuery Foundation, Inc. [ctb, cph].

shinybootstrap2 Bootstrap 2 Web Components for Use with **shiny**. Authors: Winston Chang [aut, cre], RStudio [cph], Mark Otto [ctb], Jacob Thornton [ctb], Bootstrap contributors [ctb] (Bootstrap library), Twitter, Inc [cph], Brian Reavis [ctb, cph], Egor Khmelev [ctb, cph], SpryMedia Limited [ctb, cph].

shinydashboard Create Dashboards with **shiny**. Authors: Winston Chang [aut, cre], RStudio [cph], Almasaeed Studio [ctb, cph], Adobe Systems Incorporated [ctb, cph].

shinyjs Perform Common JavaScript Operations in **shiny** Apps using Plain R Code. Author: Dean Attali [aut, cre].

shinythemes Themes for **shiny**. Authors: Winston Chang [aut, cre], RStudio [cph], Thomas Park [ctb, cph], Lukasz Dziedzic [ctb, cph], Nathan Willis [ctb, cph], Google Corporation [ctb, cph], Matt McInerney [ctb, cph], Adobe Systems Incorporated [ctb, cph], Canonical Ltd [ctb, cph].

showtextdb Font Files for the **showtex** Package. Authors: Yixuan Qiu and authors of the included fonts.

signmedian.test Perform Exact Sign Test and Asymptotic Sign Test in Large Samples. Authors: Yeyun Yu and Ting Yang.

simcausal Simulating Longitudinal Data with Causal Inference Applications. Authors: Oleg Sofrygin [aut, cre], Romain Neugebauer [aut].

simpleNeural An Easy to Use Multilayer Perceptron. Authors: David Dernoncourt [aut, cre].

sivipm Sensitivity Indices with Dependent Inputs. Authors: A. Bouvier [aut], J.-P. Gauchi [aut, cre], E. Volatier [ctb].

sjmisc Miscellaneous Data Management Tools. Author: Daniel Lüdecke.

smbinning Optimal Binning for Scoring Modeling. Author: Herman R. Jopia Bonnet.

smcfcs Multiple Imputation of Covariates by Substantive Model Compatible Fully Conditional Specification. Author: Jonathan Bartlett [aut, cre].

smds Symbolic Multidimensional Scaling. Authors: Yoshikazu Terada, Patrick J. F. Groenen.

smoof Single and Multi-Objective Optimization Test Functions. Author: Jakob Bossek [aut, cre].

solarius An R Interface to SOLAR. Authors: Andrey Ziyatdinov [cre, aut], Helena Brunel [aut], Angel Martinez-Perez [aut], Alfonso Buil [aut], Alexandre Perera [cph], Jose Manuel Soria [cph].

spTest Nonparametric Hypothesis Tests of Isotropy and Symmetry. Author: Zachary Weller [aut, cre].

spatialEco Spatial Analysis and Modeling. Authors: Jeffrey S. Evans [aut, cre], Karthik Ram [ctb].

speaq Tools for Nuclear Magnetic Resonance Spectrum Alignment and Quantitative Analysis. Authors: Trung Nghia Vu, Kris Laukens and Dirk Valkenborg. In view: *ChemPhys*.

spgs Statistical Patterns in Genomic Sequences. Authors: Andrew Hart [aut, cre], Servet Martínez [aut], Universidad de Chile [cph], INRIA-Chile [cph].

sprex Calculate Species Richness and Extrapolation Metrics. Author: Eric Archer.

ssfa Spatial Stochastic Frontier Analysis. Authors: Elisa Fusco, Francesco Vidoli. In view: *Econometrics*.

ssizeRNA Sample Size Calculation for RNA-Seq Experimental Design. Authors: Ran Bi, Peng Liu.

sspse Estimating Hidden Population Size using Respondent Driven Sampling Data. Authors: Mark S. Handcock [aut, cre, cph], Krista J. Gile [aut, cph].

staTools Statistical Tools for Social Network Analysis. Author: Alessandro Bessi.

stackoverflow Stack Overflow's Greatest Hits. Authors: Neal Fultz and the StackOverflow.com community.

stagePop Modelling the Population Dynamics of a Stage-Structured Species in Continuous Time. Author: Helen Kettle.

stcm Tools for Inference with Set-Theoretic Comparative Methods. Author: Chris Krogslund [aut, cre].

stepR Fitting Step Functions. Authors: Thomas Hotz [aut, cre], Hannes Sieling [aut], Pein Florian [ctb].

stheoreme Klimontovich's S-Theorem Algorithm Implementation and Data Preparation Tools. Author: Vitaly Efremov.

stmCorrViz A Tool for Structural Topic Model Visualizations. Authors: Antonio Coppola [aut, cre, cph], Margaret Roberts [ctb, cph], Brandon Stewart [aut, cph], Dustin Tingley [ctb, ths, cph].

stocks Fast Functions for Stock Market Analysis. Author: Dane R. Van Domelen.

supcluster Supervised Cluster Analysis. Authors: David A. Schoenfeld, Jesse Hsu.

survRM2 Comparing Restricted Mean Survival Time. Authors: Hajime Uno, Lu Tian, Angel Cronin, Chakib Battioui.

synRNASeqNet Synthetic RNA-Seq Network Generation and Mutual Information Estimates. Authors: Luciano Garofano, Stefano Maria Pagnotta, Michele Ceccarelli.

systemicrisk A Toolbox for Systemic Risk. Authors: Axel Gandy and Luitgard A.M. Veraart.

syuzhet Extracts Sentiment and Sentiment-Derived Plot Arcs from Text. Author: Matthew Jockers [aut, cre].

tdr Target Diagram. Author: Oscar Perpinan Lamigueiro [cre, aut].

texmexseq Treatment Effect eXplorer for Microbial Ecology eXperiments (using Sequence Counts). Author: Scott Olesen.

threejs Interactive 3D Scatter Plots and Globes. Author: B. W. Lewis.

thsls Three-Stage Least Squares Estimation for Systems of Simultaneous Equations. Author: Zaghdoudi Taha.

tidyjson A Grammar for Turning JSON into Tidy Tables. Author: Jeremy Stanley.

timeSeq Nonparallel Differential Expressed Genes. Authors: Fan Gao, Xiaoxiao Sun.

timetree Interface to the TimeTree of Life Webpage. Author: Franz-Sebastian Krah.

toOrdinal Function for Converting Cardinal to Ordinal Numbers by Adding a Language Specific Ordinal Indicator to the Number. Author: Damian W. Betebenner.

treeClust Cluster Distances Through Trees. Author: Sam Buttrey.

treeperm Exact and Asymptotic K Sample Permutation Test. Author: Qiao Kang.

trend Non-Parametric Trend Tests and Change-Point Detection. Author: Thorsten Pohlert. In view: *TimeSeries*.

tsallisqexp Tsallis q-Exp Distribution. Authors: Cosma Shalizi [aut], Christophe Dutang [cre]. In view: *Distributions*.

tscount Analysis of Count Time Series. Authors: Tobias Liboschik [aut, cre], Roland Fried [aut], Konstantinos Fokianos [aut], Philipp Probst [aut], Jonathan Rathjens [ctb]. In view: *TimeSeries*.

uniqtag Abbreviate Strings to Short, Unique Identifiers. Author: Shaun Jackman [cre].

vmsbase GUI Tools to Process, Analyze and Plot Fisheries Data. Authors: Lorenzo D'Andrea, Tommaso Russo, Antonio Parisi, Stefano Cataudella.

waffle Create Waffle Chart Visualizations in R. Author: Bob Rudis.

wahc Autocorrelation and Heteroskedasticity Correction in Fixed Effect Panel Data Model. Author: Zaghdoudi Taha. In view: *Econometrics*.

webchem Chemical Information from the Web. Author: Eduard Szoecs.

wkb Convert Between Spatial Objects and Well-Known Binary Geometry. Author: TIBCO Software Inc. In view: *Spatial*.

wmlf Wavelet Leaders in Multifractal Analysis. Authors: Stephane Roux, Francois Semecurbe, Cecile Tannier.

x.ent eXtraction of ENTity. Authors: Nicolas Turenne [aut], Tien T. Phan [aut, cre], John Resig [ctb, cph], Jeroen Ooms [ctb].

xml2 Parse XML. Authors: Hadley Wickham [aut, cre], Jeroen Ooms [ctb], RStudio [cph], R Foundation [ctb].

zooaRch Analytical Tools for Zooarchaeological Data. Authors: Erik Otarola-Castillo, Jesse Wolfhagen, Max D. Price.

Other changes

The following packages were moved to the Archive: **AdapEnetClass**, **BAS**, **EMDomics**, **HTSDiff**, **MGLM**, **MsatAllele**, **NCBI2R**, **NRAIA**, **PRISMA**, **PenLNM**, **PoMoS**, **Rcell**, **RfmriVC**, **TR8**, **TSgetSymbol**, **TShistQuote**, **TSjson**, **TSxls**, **TSzip**, **UScensus2000blkgrp**, **VarEff**, **bamboo**, **bark**, **bcool**, **clusthaplo**, **convexHaz**, **epicalc**, **gemmR**, **ggHorizon**, **hypred**, **imputeYn**, **integrOmics**, **jackknifeKME**, **jvmr**, **mfblock**, **miP**, **mobForest**, **muscle**, **nlrwr**, **npRmpi**, **opm**, **opmdata**, **papeR**, **parfm**, **pkgutils**, **powerr**, **pxweb**, **qp**, **qtlbim**, **rbiouml**, **rgrs**, **rsnps**, **rsprng**, **saps**, **spectral.methods**, **toaster**, **trackObjs**, **visova**, **wikipediatrend**, **xgboost**.

The following packages were resurrected from the Archive: **ARTIVA**, **AncestryMapper**, **BLCOP**, **BOG**, **IFP**, **MortalitySmooth**, **RDS**, **RSocrata**, **Rdsdp**, **TAQMNGR**, **UBCRM**, **assist**, **bigml**, **blockTools**, **eco**, **emg**, **emplik2**, **faisalconjoint**, **fracprolif**, **gammSlice**, **glrt**, **hddtools**, **hgm**, **knncat**, **longclust**, **neuRosim**, **noia**, **orthogonalsplinebasis**, **pamctdp**, **ppiPre**, **ppmlasso**, **protoclust**, **sra**, **ssize.fdr**, **survJamda**, **survJamda.data**, **tmg**, **treelet**.

The following packages had to be removed: **Rniftilib**.

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Achim Zeileis
Universität Innsbruck, Austria
Achim.Zeileis@R-project.org

Changes in R

From version 3.1.3 to version 3.2.1

by the R Core Team

CHANGES IN R 3.2.1

NEW FEATURES

- `utf8ToInt()` now checks that its input is valid UTF-8 and returns `NA` if it is not.
- `install.packages()` now allows `type = "both"` with `repos = NULL` if it can infer the type of file.
- `nchar(x,*)` and `nzchar(x)` gain a new argument `keepNA` which governs how the result for `NA`s in `x` is determined. For the R 3.2.x series, the default remains `FALSE` which is fully back compatible. From R 3.3.0, the default will change to `keepNA = NA` and you are advised to consider this for code portability.
- `news()` more flexibly extracts dates from package 'NEWS.Rd' files.
- `lengths(x)` now also works (trivially) for atomic `x` and hence can be used more generally as an efficient replacement of `sapply(x,length)` and similar.
- The included version of PCRE has been updated to 8.37, a bug-fix release.
- `diag()` no longer duplicates a matrix when extracting its diagonal.
- `as.character.srcref()` gains an argument to allow characters corresponding to a range of source references to be extracted.

BUG FIXES

- `acf()` and `ccf()` now guarantee values strictly in $[-1, 1]$ (instead of sometimes very slightly outside). PR#15832.
- `as.integer("111111111111")` now gives NA (with a warning) as it does for the corresponding numeric or negative number coercions. Further, `as.integer(M + 0.1)` now gives `M` (instead of NA) when M is the maximal representable integer.
- On some platforms `nchar(x,"c")` and `nchar(x,"w")` would return values (possibly `NA`) for inputs which were declared to be UTF-8 but were not, or for invalid strings without a marked encoding in a multi-byte locale, rather than give an error. Additional checks have been added to mitigate this.
- `apply(a,M,function(u) c(X = .,Y = .))` again has dimnames containing "X" and "Y" (as in R < 3.2.0).
- (Windows only) In some cases, the `--clean` option to `R CMD INSTALL` could fail. (PR#16178)
- (Windows only) `choose.files()` would occasionally include characters from the result of an earlier call in the result of a later one. (PR#16270)
- A change in `RSiteSearch()` in R 3.2.0 caused it to submit invalid URLs. (PR#16329)
- `Rscript` and command line `R` silently ignored incomplete statements at the end of a script; now they are reported as parse errors. (PR#16350)
- Parse data for very long strings was not stored. (PR#16354)

- plotNode(), the workhorse of the plot method for "dendrogram"s is no longer recursive, thanks to Suharto Anggono, and hence also works for deeply nested dendrograms. (PR#15215)
- The parser could overflow internally when given numbers in scientific format with extremely large exponents. (PR#16358)
- If the CRAN mirror was not set, install.packages(type = "both") and related functions could repeatedly query the user for it. (Part of PR#16362)
- The low-level functions .rowSums() etc. did not check the length of their argument, so could segfault. (PR#16367)
- The quietly argument of library() is now correctly propagated from .getRequiredPackages2().
- Under some circumstances using the internal PCRE when building R fron source would cause external libs such as -llzma to be omitted from the main link.
- The .Primitive default methods of the logic operators, i.e., !, & and |, now give correct error messages when appropriate, e.g., for \`&\`(TRUE) or \`!\`(). (PR#16385)
- cummax(x) now correctly propagates NAs also when x is of type integer and begins with an NA.
- summaryRprof() could fail when the profile contained only two records. (PR#16395)
- HTML vignettes opened using vignette() did not support links into the rest of the HTML help system. (Links worked properly when the vignette was opened using browseVignettes() or from within the help system.)
- arima(*,xreg = .) (for $d \geq 1$) computes estimated variances based on a the number of effective observations as in R version 3.0.1 and earlier. (PR#16278)
- slotNames(.) is now correct for "signature" objects (mostly used internally in **methods**).
- On some systems, the first string comparison after a locale change would result in NA.

CHANGES IN R 3.2.0

NEW FEATURES

- anyNA() gains a recursive argument.
- When x is missing and names is not false (including the default value), Sys.getenv(x,names) returns an object of class "Dlist" and hence prints tidily.
- (Windows.) shell() no longer consults the environment variable SHELL: too many systems have been encountered where it was set incorrectly (usually to a path where software was compiled, not where it was installed). R_SHELL, the preferred way to select a non-default shell, can be used instead.
- Some unusual arguments to embedFonts() can now be specified as character vectors, and the defaults have been changed accordingly.
- Functions in the Summary group duplicate less. (PR#15798)
- (Unix-alikes.) system(cmd,input =) now uses 'shell-execution-environment' redirection, which will be more natural if cmd is not a single command (but requires a POSIX-compliant shell). (Wish of PR#15508)
- read.fwf() and read.DIF() gain a fileEncoding argument, for convenience.

- Graphics devices can add attributes to their description in `.Device` and `.Devices`. Several of those included with R use a `"filepath"` attribute.
- `pmatch()` uses hashing in more cases and so is faster at the expense of using more memory. (PR#15697)
- `pairs()` gains new arguments to select sets of variables to be plotted against each other.
- `file.info(,extra_cols = FALSE)` allows a minimal set of columns to be computed on Unix-alikes: on some systems without properly-configured caching this can be significantly faster with large file lists.
- New function `dir.exists()` in package **base** to test efficiently whether one or more paths exist and are directories.
- `dput()` and friends gain new controls '`hexNumeric`' and '`digits17`' which output double and complex quantities as, respectively, binary fractions (exactly, see `sprintf("%a")`) and as decimals with up to 17 significant digits.
- `save()`, `saveRDS()` and `serialize()` now support `ascii = NA` which writes ASCII files using `sprintf("%a")` for double/complex quantities. This is read-compatible with `ascii = TRUE` but avoids binary->decimal->binary conversions with potential loss of precision. Unfortunately the Windows C runtime's lack of C99 compliance means that the format cannot be read correctly there in R before 3.1.2.
- The default for `formatC(decimal.mark =)` has been changed to be `getOption("OutDec")`; this makes it more consistent with `format()` and suitable for use in print methods, e.g. those for classes `"density"`, `"ecdf"`, `"stepfun"` and `"summary.lm"`.

  `getOption("OutDec")` is now consulted by the print method for class `"kmeans"`, by `cut()`, `dendrogram()`, `plot.ts()` and `quantile()` when constructing labels and for the report from `legend(trace = TRUE)`.

  (In part, wish of PR#15819.)
- `printNum()` and hence `format()` and `formatC()` give a warning if `big.mark` and `decimal.mark` are set to the same value (period and comma are not uncommonly used for each, and this is a check that conventions have not got mixed).
- `merge()` can create a result which uses long vectors on 64-bit platforms.
- `dget()` gains a new argument `keep.source` which defaults to `FALSE` for speed (`dput()` and `dget()` are most often used for data objects where this can make `dget()` many times faster).
- Packages may now use a file of common macro definitions in their help files, and may import definitions from other packages.
- A number of macros have been added in the new '`share/Rd`' directory for use in package overview help pages, and `promptPackage()` now makes use of them.
- `tools::parse_Rd()` gains a new `permissive` argument which converts unrecognized macros into text. This is used by `utils:::format.bibentry` to allow LaTeX markup to be ignored.
- `options(OutDec =)` can now specify a multi-byte character, e.g., `options(OutDec = "\u00b7")` in a UTF-8 locale.
- `is.recursive(x)` is no longer true when `x` is an external pointer, a weak reference or byte code; the first enables `all.equal(x,x)` when `x <-getClass(.)`.
- `ls()` (aka `objects()`) and `as.list.environment()` gain a new argument `sorted`.

- The `"source"` attribute (which has not been added to functions by R since before R version 2.14.0) is no longer treated as special.
- Function `returnValue()` has been added to give `on.exit()` code access to a function's return value for debugging purposes.
- `crossprod(x,y)` allows more matrix coercions when `x` or `y` are vectors, now equalling `t(x) %*% y` in these cases (also reported by Radford Neal). Similarly, `tcrossprod(x,y)` and `%*%` work in more cases with vector arguments.
- Utility function `dynGet()` useful for detecting cycles, aka infinite recursions.
- The byte-code compiler and interpreter include new instructions that allow many scalar subsetting and assignment and scalar arithmetic operations to be handled more efficiently. This can result in significant performance improvements in scalar numerical code.
- `apply(m,2,identity)` is now the same as the matrix `m` when it has *named* row names.
- A new function `debuggingState()` has been added, allowing to temporarily turn off debugging.
- `example()` gets a new optional argument `run.donttest` and `tools::Rd2ex()` a corresponding `commentDonttest`, with a default such that `example(..)` in help examples will run `\donttest` code only if used interactively (a change in behaviour).
- `rbind.data.frame()` gains an optional argument `make.row.names`, for potential speedup.
- New function `extSoftVersion()` to report on the versions of third-party software in use in this session. Currently reports versions of `zlib`, `bzlib`, the `liblzma` from `xz`, PCRE, ICU, TRE and the `iconv` implementation.

  A similar function `grSoftVersion()` in package **grDevices** reports on third-party graphics software.

  Function `tcltk::tclVersion()` reports the Tcl/Tk version.
- Calling `callGeneric()` without arguments now works with primitive generics to some extent.
- `vapply(x,FUN,FUN.VALUE)` is more efficient notably for large `length(FUN.VALUE)`; as extension of PR#16061.
- `as.table()` now allows tables with one or more dimensions of length 0 (such as `as.table(integer())`).
- `names(x) <-NULL` now clears the names of call and `...` objects.
- `library()` will report a warning when an insufficient dependency version is masking a sufficient one later on the library search path.
- A new `plot()` method for class `"raster"` has been added.
- New `check_packages_in_dir_changes()` function in package **tools** for conveniently analyzing how changing sources impacts the check results of their reverse dependencies.
- Speed-up from Peter Haverty for `ls()` and `methods:::.requirePackage()` speeding up package loading. (PR#16133)
- New `get0()` function, combining `exists()` and `get()` in one call, for efficiency.
- `match.call()` gains an `envir` argument for specifying the environment from which to retrieve the `...` in the call, if any; this environment was wrong (or at least undesirable) when the `definition` argument was a function.

- `topenv()` has been made `.Internal()` for speedup, based on Peter Haverty's proposal in PR#16140.
- `getOption()` no longer calls `options()` in the main case.
- Optional use of `libcurl` (version 7.28.0 from Oct 2012 or later) for Internet access:
  - `capabilities("libcurl")` reports if this is available.
  - `libcurlVersion()` reports the version in use, and other details of the `"libcurl"` build including which URL schemes it supports.
  - `curlGetHeaders()` retrieves the headers for `http://`, `https://`, `ftp://` and `ftps://` URLs: analysis of these headers can provide insights into the 'existence' of a URL (it might for example be permanently redirected) and is so used in `R CMD check --as-cran`.
  - `download.file()` has a new optional method `"libcurl"` which will handle more URL schemes, follow redirections, and allows simultaneous downloads of multiple URLs.
  - `url()` has a new method `"libcurl"` which handles more URL schemes and follows redirections. The default method is controlled by a new option `url.method`, which applies also to the opening of URLs *via* `file()` (which happens implicitly in functions such as `read.table`.)
  - When `file()` or `url()` is invoked with a `https://` or `ftps://` URL which the current method cannot handle, it switches to a suitable method if one is available.
- (Windows.) The DLLs 'internet.dll' and 'internet2.dll' have been merged. In this version it is safe to switch (repeatedly) between the internal and Windows internet functions within an R session.

  The Windows internet functions are still selected by flag '`--internet2`' or `setInternet2()`. This can be overridden for an `url()` connection *via* its new `method` argument.

  `download.file()` has new method `"wininet"`, selected as the default by '`--internet2`' or `setInternet2()`.
- `parent.env<-` can no longer modify the parent of a locked namespace or namespace imports environment. Contributed by Karl Millar.
- New function `isNamespaceLoaded()` for readability and speed.
- `names(env)` now returns all the object names of an `environment` `env`, equivalently to `ls(env,all.names = TRUE,sorted = FALSE)` and also to the names of the corresponding list, `names(as.list(env,all.names = TRUE))`. Note that although `names()` returns a character vector, the names have no particular ordering.
- The memory manager now grows the heap more aggressively. This reduces the number of garbage collections, in particular while data or code are loaded, at the expense of slightly increasing the memory footprint.
- New function `trimws()` for removing leading/trailing whitespace.
- `cbind()` and `rbind()` now consider S4 inheritance during S3 dispatch and also obey `deparse.level`.
- `cbind()` and `rbind()` will delegate recursively to `methods::cbind2` (`methods::rbind2`) when at least one argument is an S4 object and S3 dispatch fails (due to ambiguity).
- (Windows.) `download.file(quiet = FALSE)` now uses text rather than Windows progress bars in non-interactive use.

- New function `hsearch_db()` in package **utils** for building and retrieving the help search database used by `help.search()`, along with functions for inspecting the concepts and keywords in the help search database.
- New function `.getNamespaceInfo()`, a no-check version of `getNamespaceInfo()` mostly for internal speedups.
- The help search system now takes '`\keyword`' entries in Rd files which are not standard keywords (as given in 'KEYWORDS' in the R documentation directory) as concepts. For standard keyword entries the corresponding descriptions are additionally taken as concepts.
- New `lengths()` function for getting the lengths of all elements in a list.
- New function `toTitleCase()` in package **tools**, tailored to package titles.
- The matrix methods of `cbind()` and `rbind()` allow matrices as inputs which have 2^{31} or more elements. (For `cbind()`, wish of PR#16198.)
- The default method of `image()` has an explicit check for a numeric or logical matrix (which was always required).
- `URLencode()` will not by default encode further URLs which appear to be already encoded.
- `BIC(mod)` and `BIC(mod,mod2)` now give non-NA numbers for `arima()` fitted models, as `nobs(mod)` now gives the number of "used" observations for such models. This fixes PR#16198, quite differently than proposed there.
- The `print()` methods for `"htest"`, `"pairwise.htest"` and `"power.htest"` objects now have a `digits` argument defaulting to (a function of) `getOption("digits")`, and influencing all printed numbers coherently. Unavoidably, this changes the display of such test results in some cases.
- Code completion for namespaces now recognizes all loaded namespaces, rather than only the ones that are also attached.
- The code completion mechanism can now be replaced by a user-specified completer function, for (temporary) situations where the usual code completion is inappropriate.
- `unzip()` will now warn if it is able to detect truncation when unpacking a file of 4GB or more (related to PR#16243).
- `methods()` reports S4 in addition to S3 methods; output is simplified when the `class` argument is used. `.S3methods()` and `methods:::.S4methods()` report S3 and S4 methods separately.
- Higher order functions such as the `apply` functions and `Reduce()` now force arguments to the functions they apply in order to eliminate undesirable interactions between lazy evaluation and variable capture in closures. This resolves PR#16093.

INSTALLATION and INCLUDED SOFTWARE

- The `\donttest` sections of R's help files can be tested by
  `make check TEST_DONTTEST=TRUE` .
- It is possible to request the use of system `valgrind` headers *via* `configure` option '`--with-system-valgrind-headers`': note the possible future incompatibility of such headers discussed in the 'R Installation and Administration' manual. (Wish of PR#16068.)
- The included version of `liblzma` has been updated to `xz-utils` 5.0.7 (minor bug fixes from 5.0.5).

- `configure` options '`--with-system-zlib`', '`--with-system-bzlib`' and '`--with-system-pcre`' are now the default. For the time being there is fallback to the versions included in the R sources if no system versions are found or (unlikely) if they are too old.

  Linux users should check that the `-devel` or `-dev` versions of packages **zlib**, **bzip2**/**libbz2** and **pcre** as well as **xz-devel**/**liblzma-dev** (or similar names) are installed.

- `configure` by default looks for the `texi2any` script from **texinfo** 5.1 or later, rather than the `makeinfo` program. (`makeinfo` is a link to the Perl script `texi2any` in **texinfo** 5.x.)

- `R CMD INSTALL` gains an option '`--built-timestamp=STAMP`' allowing 100% reproducible package building, thanks to Dirk Eddelbuettel.

UTILITIES

- There is support for testing the `\dontrun` and `\donttest` parts of examples in packages.

  `tools::testInstalledPackage()` accepts new arguments `commentDontrun = FALSE` and `commentDonttest = FALSE`.

  `R CMD check` gains options '`--run-dontrun`' and '`--run-donttest`'.

- The HTML generated by `tools::Rd2HTML()` and `tools::toHTML()` methods is now 'XHTML 1.0 Strict'.

- The **compiler** package's utility function `setCompilerOptions()` now returns the old values invisibly. The initial optimization level can also be set with the environment variable `R_COMPILER_OPTIMIZE`.

- `R CMD build` adds a '`NeedsCompilation`' field if one is not already present in the 'DESCRIPTION' file.

- `R CMD check` gains option '`--test-dir`' to specify an alternative set of tests to run.

- `R CMD check` will now by default continue with testing after many types of errors, and will output a summary count of errors at the end if any have occurred.

- `R CMD check` now checks that the '`Title`' and '`Description`' fields are correctly terminated.

- `R CMD check --as-cran` now:

  - checks a 'README.md' file can be processed: this needs pandoc installed.
  - checks the existence and accessibility of URLs in the 'DESCRIPTION', 'CITATION', 'NEWS.Rd' and 'README.md' files and in the help files (provided the build has `libcurl` support).
  - reports non-ASCII characters in R source files when there is no package encoding declared in the 'DESCRIPTION' file.
  - reports (apparent) S3 methods exported but not registered.
  - reports overwriting registered S3 methods from base/recommended packages. (Such methods are replaced in the affected package for the rest of the session, even if the replacing namespace is unloaded.)
  - reports if the `Title` field does not appear to be in title case (see 'Writing R Extensions': there may be false positives, but note that technical words should be single-quoted and will then be accepted).

  Most of these checks can also be selected by environment variables: see the 'R Internals' manual.

C-LEVEL FACILITIES

- New C API utility `logspace_sum(logx[],n)`.
- Entry points `rbinom_mu`, `rnbinom_mu` and `rmultinom` are remapped (by default) to `Rf_rbinom_mu` etc. This requires packages using them to be re-installed.
- `.C(DUP = FALSE)` and `.Fortran(DUP = FALSE)` are now ignored, so arguments are duplicated if `DUP = TRUE` would do so. As their help has long said, `.Call()` is much preferred.
- New entry point `R_allocLD`, like `R_alloc` but guaranteed to have sufficient alignment for `long double` pointers.
- `isPairList()` now returns `TRUE` for DOTSXP.

WINDOWS BUILD CHANGES

A number of changes to the Windows build system are in development. The following are currently in place.

- Installation using external binary distributions of **zlib**, **bzip2**, **liblzma**, **pcre**, **libpng**, **jpeglib** and **libtiff** is now required, and the build instructions have been revised.
- A new `make` target `rsync-extsoft` has been added to obtain copies of the external libraries from CRAN.
- Building the manuals now requires `texi2any` from **texinfo** 5.1 or later. CRAN binary builds include the manuals, but by default builds from source will not, and they will be accessed from CRAN. See the comments in 'src/gnuwin32/MkRules.dist' for how to specify the location of `texi2any`.
- (Windows) Changes have been made to support an experimental Windows toolchain based on GCC 4.9.2. The default toolchain continues to be based on GCC 4.6.3, as the new toolchain is not yet stable enough. A change to a new toolchain is expected during the R 3.2.x lifetime.

PACKAGE INSTALLATION

- (Windows) The use of macro `ZLIB_LIBS` in file 'src/Makevars.win' (which has not been documented for a long time) now requires an external 'libz.a' to be available (it is part of the 'goodies' used to compile Windows binary packages). It would be simpler to use `-lz` instead.
- The default for option `pkgType` on platforms using binary packages is now `"both"`, so source packages will be tried if binary versions are not available or not up to date.

  There are options for what `install.packages(type = "both")` (possibly called *via* `update.packages()`) will do if compilation of a source package is desirable: see `?options` (under **utils**).

  If you intend not to accept updates as source packages, you should use `update.packages(type = "binary")`.

DEPRECATED AND DEFUNCT

- `download.file(method = "lynx")` is defunct.
- Building R using the included versions of `zlib`, `bzip2`, `xz` and PCRE is deprecated: these are frozen (bar essential bug-fixes) and will be removed for R 3.3.0.

- The configure option '--with-valgrind-instrumentation=3' has been withdrawn, as it did not work with recent valgrind headers: it is now treated as level 2.
- The MethodsList class in package **methods** had been deprecated in R 2.11.0 and is defunct now. Functions using it are defunct if they had been deprecated in R 2.11.0, and are deprecated now, otherwise.

BUG FIXES

- Fixed two obscure bugs in pairlist subassignment, reported by Radford Neal as part of pqR issue 16.
- Fixes for bugs in handling empty arguments and argument matching by name in log().
- all.equal() gains methods for environments and refClasses.
- [<- and [[<- gain S4 data.frame methods to avoid corruption of S4 class information by the S3 methods.
- callNextMethod() should now work within a .local call when ... is absent from formals(.local).
- dput(pairlist(x)) generates a call to the pairlist constructor instead of the list constructor.
- Fix missing() when arguments are propagated through .... (PR#15707)
- eigen(m) now defaults to symmetric = TRUE even when the dimnames are asymmetric if the matrix is otherwise symmetric. (PR#16151)
- Fix issues with forwarding ... through callGeneric() and callNextMethod(). (PR#16141)
- callGeneric() now works after a callNextMethod().
- Subclass information is kept consistent when replacing an ordinary S4 class with an "old class" *via* the S4Class argument to setOldClass(). Thus, for example, a data.frame is valid for a list argument in the signature, and a factor is valid for vector arguments.
- In qbeta() the inversion of pbeta() is much more sophisticated. This works better in corner cases some of which failed completely previously (PR#15755), or were using too many iterations.
- Auto-printing no longer duplicates objects when printing is dispatched to a method.
- kmeans(x,k) would fail when nrow(x) >= 42949673. (Comment 6 of PR#15364)
- 'Abbreviated' locale-specific day and month names could have been truncated in those rare locales where there are the same as the full names.
- An irrelevant warning message from updating subclass information was silenced (the namespace would not be writable in this case).

CHANGES IN R 3.1.3

NEW FEATURES

- The internal method of download.file() can now handle files larger than 2GB on 32-bit builds which support such files (tested on 32-bit R running on 64-bit Windows).

- kruskal.test() warns on more types of suspicious input.
- The as.dendrogram() method for "hclust" objects gains a check argument protecting against memory explosion for invalid inputs.
- capabilities() has a new item long.double which indicates if the build uses a long double type which is longer than double.
- nlm() no longer modifies the callback argument in place (a new vector is allocated for each invocation, which mimics the implicit duplication that occurred in R < 3.1.0); note that this is a change from the previously documented behavior. (PR#15958)
- icuSetCollate() now accepts locale = "ASCII" which uses the basic C function strcmp and so collates strings byte-by-byte in numerical order.
- sessionInfo() tries to report the OS version in use (not just that compiled under, and including details of Linux distributions).
- model.frame() (used by lm() and many other modelling functions) now warns when it drops contrasts from factors. (Wish of PR#16119)
- install.packages() and friends now accept the value type = "binary" as a synonym for the native binary type on the platform (if it has one).
- Single source or binary files can be supplied for install.packages(type = "both") and the appropriate type and repos = NULL will be inferred.
- New function pcre_config() to report on some of the configuration options of the version of PCRE in use. In particular, this reports if regular expressions using '\p{xx}' are supported.
- (Windows.) download.file(cacheOK = FALSE) is now supported when 'internet2.dll' is used.
- browseURL() has been updated to work with Firefox 36.0 which has dropped support for the '-remote' interface.

INSTALLATION and INCLUDED SOFTWARE

- The included version of PCRE has been updated to 8.36.
- configure accepts 'MAKEINFO=texi2any' as another way to ensure **texinfo** 5.x is used when both 5.x and 4.x are installed.

UTILITIES

- R CMD check now checks the packages used in \donttest sections of the examples are specified in the 'DESCRIPTION' file. (These are needed to run the examples interactively.)
- R CMD check checks for the undeclared use of GNU extensions in Makefiles, and for Makefiles with a missing final linefeed.

  R CMD build will correct line endings in all Makefiles, not just those in the 'src' directory.
- R CMD check notes uses of library() and require() in package code: see the section 'Suggested packages' of 'Writing R Extensions' for good practice.

DEPRECATED AND DEFUNCT

- The configure option '--with-valgrind-instrumentation=3' is deprecated and will be removed in R 3.2.0.

BUG FIXES

- (Windows.) Rscript.exe was missing a manifest specifying the modern style for common controls (e.g., the download progress bar).
- If a package had extra documentation files but no vignette, the HTML help system produced an empty index page.
- The parser now gives an error if a null character is included in a string using Unicode escapes. (PR#16046)
- qr.Q() failed on complex arguments due to pre-3.0(!) typo. (PR#16054)
- abs() failed with named arguments when the argument was complex. (PR#16047)
- "noquote" objects may now be used as columns in dataframes. (PR#15997)
- Some values with extremely long names were printed incorrectly. (PR#15999)
- Extremely large exponents on zero expressed in scientific notation (e.g. 0.0e50000) could give NaN. (PR#15976)
- download.file() reported downloaded sizes as 0KB if less than 1MB, only for R 3.1.2 and only on big-endian platforms.
- prompt() did not escape percent signs in the automatically generated usage section of help files.
- drop.terms() dropped some of the attributes of the object it was working with. (PR#16029)
- (Windows.) The command completion in Rgui.exe messed up the console. (PR#15791)
- (Windows.) The choose.files() command returned a blank string when the user asked for a single file but cancelled the request. (PR#16074)
- Math2 S4 group generics failed to correctly dispatch "structure"- and "nonStructure"-derived classes.
- loadNamespace() imposed undocumented restrictions on the versionCheck parameter. (Reported by Geoff Lee.)
- Rare over-runs detected by AddressSanitizer in substr() and its replacement version have been avoided.

  Inter alia that fix gives the documented behaviour for substr(x,1,2) <-"" (subsequently reported as PR#16214).
- Loading packages incorrectly defining an S4 generic followed by a function of the same name caused an erroneous cyclic namespace dependency error.
- Declared vignette encodings are now always passed to the vignette engine.
- Port Tomas Kalibera's fix from R-devel that restores the loadMethod() fast path, effectively doubling the speed of S4 dispatch.
- power.t.test() and power.prop.test() now make use of the extendInt option of uniroot() and hence work in more extreme cases. (PR#15792)
- If a package was updated and attached when its namespace was already loaded, it could end up with parts from one version and parts from the other. (PR#16120)
- tools:::.Rdconv() didn't accept --encoding= due to a typo. (PR#16121)
- Unix-alike builds without a suitable makeinfo were documented to link the missing HTML manuals to CRAN, but did not.

- `save(*,ascii=TRUE)` and `load()` now correctly deal with `NaN`'s. (PR#16137)
- `split.Date()` retains fractional representations while avoiding incomplete class propagation.
- 'R_ext/Lapack.h' had not been updated for changes made by LAPACK to the argument lists of its (largely internal) functions `dlaed2` and `dlaed3`. (PR#16157)
- `RShowDoc("NEWS","txt")` had not been updated for the layout changes of R 3.1.0.
- The `xtfrm()` method for class `"Surv"` has been corrected and its description expanded.
- `mode(x) <-y` would incorrectly evaluate `x` before changing its mode. (PR#16215)
- `besselJ(1,2^64)` and `besselY(..)` now signal a warning, returning `NaN` instead of typically segfaulting. (Issue 3 of PR#15554)
- HTML conversion of '`\href`' markup in '.Rd' files did not remove the backslash from '`\%`' and so gave an invalid URL. In a related change, the '`\`' escape is now required in such URLs.

News from the Bioconductor Project

by the Bioconductor Team

The Bioconductor project provides tools for the analysis and comprehension of high-throughput genomic data. The 1024 software packages available in Bioconductor can be viewed at `http://bioconductor.org/packages/`. Navigate packages using 'biocViews' terms and title search. Each package has an html page with a description, links to vignettes, reference manuals, and usage statistics. Start using Bioconductor version 3.1 by installing R 3.2.1 and evaluating the commands

```
source("http://bioconductor.org/biocLite.R")
biocLite()
```

Install additional packages and dependencies, e.g., **AnnotationHub**, with

```
source("http://bioconductor.org/biocLite.R")
biocLite("AnnotationHub")
```

Bioconductor 3.1 release highlights

Bioconductor 3.1 was released on 17 April 2015. It is compatible with R 3.2 and consists of 1024 software packages, 214 experiment data packages, and more 917 up-to-date annotation packages. There are 95 new software packages and many updates and improvements to existing packages. The release announcement includes descriptions of new packages and updated NEWS files provided by package maintainers. There are a great diversity of packages represented. Highlights include: **sincell** for assessment of cell-state hierarchies from single-cell data; gene set and network analysis packages (e.g., **RCyjs mogsa**, **nethet**, **pandaR**, **pwOmics**, **seq2pathway**); packages for methylation (e.g., **BEclear**, **conumee**, **DMRcaller**, **ENmix**, **RnBeads**, **skewr**), flow cytometry (e.g., **FlowRepositoryR**, **FlowSOM**, **flowVS**, **immunoClus**), and other domain-specific analysis (e.g., **LEA**, for landscape genomics); access to multiple-sequence alignment algorithms (**muscle**, **msa**); visualization packages such as **gtrellis** (genome level Trellis graph visualizes), **ggtree** (phylogenetic tree and associated annotation data), **ComplexHeatmap**, **seqPattern** (oligonucleotide patterns and sequence motifs centred at a common reference point), and **soGGI** (genomic interval aggregate and summary plots of signal or motif occurrence); and infrastructure packages such as **Rhtslib** (wrapping a recent version of the htslib C library for processing CRAM, BAM, and other high-throughput sequence files) and **GoogleGenomics** (to interact with the Google Genomics interface).

Our collection of microarray, transcriptome and organism-specific *annotation packages* use the 'select' interface (keys, columns, keytypes) to access static information on gene annotations (**org.*** packages) and gene models (**TxDb.*** packages); these augment packages such as **biomaRt** for interactive querying of web-based resources and **VariantAnnotation**, **VariantFiltering**, and **ensemblVEP** for annotation of DNA sequences. Use of these resources are documented in updated annotation workflows. The **AnnotationHub** complements our traditional offerings with diverse whole genome annotations from Ensembl, ENCODE, dbSNP, UCSC, and elsewhere; a recent addition includes Roadmap Epigenomics resources, with use described in the AnnotationHub How-To vignette.

Other activities

Several enhancements to the Bioconductor web and support sites aim to help users to identify appropriate packages while encouraging developers to provide high-quality software. In addition to biocViews terms to classify packages, each 'landing' page (e.g., `http://bioconductor.org/packages/DESeq2`) contains graphical shields that highlight cross-platform availability, download percentile, support site queries and responses, years in Bioconductor, and maintenance activity during the last 6 months. Landing pages also contain shields more useful to package maintainers, including current build status and test coverage. Users posting questions on the support site are encouraged to add tags identifying the package the question is about; maintainers are subscribed to the tag and receive email notification of the question. This enables maintainers to stay informed of problems with their package, without requiring daily monitoring of the support site.

Continued availability of Bioconductor Docker and Amazon images provides a very effective on-ramp for power users to rapidly obtain access to standardized and scalable computing environments. Docker images are available for release and development versions of Bioconductor, with analysis-specific images pre-loaded with packages relevant to common analysis scenarios, e.g., of sequencing, microarray, flow cell, or proteomic data. Both Amazon and Docker images include RStudio Server for easy web-browser based access.

New Bioconductor package contributors are encouraged to consult the package guidelines and submission sections of the Bioconductor web site, and use the **BiocCheck** package, in addition to `R CMD check`, for guidance on conforming to Bioconductor package standards. New package submissions are automatically built across Linux, Mac, and Windows platforms, providing an opportunity to address cross-platform issues; many new package contributors take advantage of this facility to refine their package before it is subject to technical preview. Keep abreast of packages added to the 'devel' branch and other activities by following @Bioconductor on Twitter.

The Bioconductor web site advertises training and community events, including the BioC 2015, the Bioconductor annual conference, to be held in Seattle, 20–22 July.

The Bioconductor Team
Program in Computational Biology
Fred Hutchinson Cancer Research Center