The R Journal

Volume 5/1, June 2013

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

Contents

The R Journal is a peer-reviewed publication of the R Foundation for Statistical Computing. Communications regarding this publication should be addressed to the editors.

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

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

Email of editors and editorial board:
firstname.lastname@R-project.org

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

Editorial

by Hadley Wickham

I'm very pleased to published my first issue of the R Journal as editor. As well as this visible indication of my working, I've also been hard at work behind the scenes to modernise some of the code that powers the R Journal. This includes:

- The new one-column style that you'll see in this issue. We changed because the R Journal is an electronic journal, and one column is much easier to view on screen. It also makes it much easier to intermingle code and text, which is so important for the R Journal. We also made a few tweaks to the typography which will hopefully make it a little more pleasant to view. Love it? Hate it? Please let me know and we may do another round of smaller tweaks for the next issue.
- I've also written an R package to manage the submission process. You won't notice this directly as a reader or author, but it should help us to keep track of articles and make sure we respond in a timely manner.
- The R Journal website is now built using jekyll, a modern static site generator. You shouldn't notice any difference, but it gives us a bit more freedom to introduce new features.

In this issue

We have a bumper crop of articles. I've roughly categorised them below.
Visualisation:

- "**osmar**: OpenStreetMap and R" by Manuel J. A. Eugster and Thomas Schlesinger
- "Let Graphics Tell the Story - Datasets in R" by Antony Unwin, Heike Hofmann and Dianne Cook
- "**ggmap**: Spatial Visualization with **ggplot2**" by David Kahle and Hadley Wickham

Text mining:

- "**RTextTools**: A Supervised Learning Package for Text Classification" by Timothy P. Jurka, Loren Collingwood, Amber E. Boydstun, Emiliano Grossman and Wouter van Atteveldt
- "**RcmdrPlugin.temis**, a Graphical Integrated Text Mining Solution in R" by Milan Bouchet-Valat and Gilles Bastin

Modelling:

- "Multiple Factor Analysis for Contingency Tables in the **FactoMineR** Package" by Belchin Kostov, Mónica Bécue-Bertaut and François Husson
- "Hypothesis Tests for Multivariate Linear Models Using the **car** Package" by John Fox, Michael Friendly and Sanford Weisberg
- "Estimating Spatial Probit Models in R" by Stefan Wilhelm and Miguel Godinho de Matos
- "Fast Pure R Implementation of GEE: Application of the **Matrix** Package" by Lee S. McDaniel, Nicholas C. Henderson and Paul J. Rathouz
- "**ftsa**: An R Package for Analyzing Functional Time Series" by Han Lin Shang

The R ecosystem:

- "Statistical Software from a Blind Person's Perspective" by A. Jonathan R. Godfrey
- "Possible Directions for Improving Dependency Versioning in R" by Jeroen Ooms
- "Translating Probability Distributions: From R to BUGS and Back Again" by David LeBauer, Michael Dietze, and Ben Bolker

And a range of papers applying R to diverse subject areas:

- "**PIN**: Measuring Asymmetric Information in Financial Markets with R" by Paolo Zagaglia
- "Generalized Simulated Annealing for Global Optimization: The **GenSA** Package" by Yang Xiang, Sylvain Gubian, Brian Suomela and Julia Hoeng
- "**QCA**: A Package for Qualitative Comparative Analysis" by Alrik Thiem and Adrian Duşa
- "An Introduction to the **EcoTroph** R Package: Analyzing Aquatic Ecosystem Trophic Networks" by Mathieu Colléter, Jérôme Guitton and Didier Gascuel
- "**stellaR**: A Package to Manage Stellar Evolution Tracks and Isochrones" by Matteo Dell'Omodarme and Giada Valle
- "**mpoly**: Multivariate Polynomials in R" by David Kahle
- "The **beadarrayFilter**: An R Package to Filter Beads", by Anyiawung Chiara Forcheh, Geert Verbeke, Adetayo Kasim, Dan Lin, Ziv Shkedy, Willem Talloen, Hinrich W.H. Goehlmann, and Lieven Clement.

Hadley Wickham, RStudio
hadley.wickham@r-project.org

RTextTools: A Supervised Learning Package for Text Classification

by Timothy P. Jurka, Loren Collingwood, Amber E. Boydstun, Emiliano Grossman, and Wouter van Atteveldt

Abstract Social scientists have long hand-labeled texts to create datasets useful for studying topics from congressional policymaking to media reporting. Many social scientists have begun to incorporate machine learning into their toolkits. **RTextTools** was designed to make machine learning accessible by providing a start-to-finish product in less than 10 steps. After installing **RTextTools**, the initial step is to generate a document term matrix. Second, a container object is created, which holds all the objects needed for further analysis. Third, users can use up to nine algorithms to train their data. Fourth, the data are classified. Fifth, the classification is summarized. Sixth, functions are available for performance evaluation. Seventh, ensemble agreement is conducted. Eighth, users can cross-validate their data. Finally, users write their data to a spreadsheet, allowing for further manual coding if required.

Introduction

The process of hand-labeling texts according to topic, tone, and other quantifiable variables has yielded datasets offering insight into questions that span social science, such as the representation of citizen priorities in national policymaking (Jones et al., 2009) and the effects of media framing on public opinion (Baumgartner et al., 2008). This process of hand-labeling, known in the field as "manual coding", is highly time consuming. To address this challenge, many social scientists have begun to incorporate machine learning into their toolkits. As computer scientists have long known, machine learning has the potential to vastly reduce the amount of time researchers spend manually coding data. Additionally, although machine learning cannot replicate the ability of a human coder to interpret the nuances of a text in context, it does allow researchers to examine systematically both texts and coding scheme performance in a way that humans cannot. Thus, the potential for heightened efficiency and perspective make machine learning an attractive approach for social scientists both with and without programming experience.

Yet despite the rising interest in machine learning and the existence of several packages in R, no package incorporates a start-to-finish product that would be appealing to those social scientists and other researchers without technical expertise in this area. We offer to fill this gap through **RTextTools**. Using a variety of existing R packages, **RTextTools** is designed as a one-stop-shop for conducting supervised learning with textual data. In this paper, we outline a nine-step process, discuss the core functions of the program, and demonstrate the use of **RTextTools** with a working example.

The core philosophy driving **RTextTools**' development is to create a package that is easy to use for individuals with no prior R experience, yet flexible enough for power users to utilize advanced techniques. Overall, **RTextTools** offers a comprehensive approach to text classification, by interfacing with existing text pre-processing routines and machine learning algorithms and by providing new analytics functions. While existing packages can be used to perform text preprocessing and machine learning, respectively, no package combines these processes with analytical functions for evaluating machine learning accuracy. In short, **RTextTools** expedites the text classification process: everything from the installation process to training and classifying has been streamlined to make machine learning with text data more accessible.

General workflow

A user starts by loading his or her data from an Access, CSV, Excel file, or series of text files using the `read_data()` function. We use a small and randomized subset of the congressional bills database constructed by Adler and Wilkerson (2004) for the running example offered here.[1] We choose this truncated dataset in order to minimize the amount of memory used, which can rapidly overwhelm a computer when using large text corpora. Most users therefore should be able to reproduce the example. However, the resulting predictions tend to improve (nonlinearly) with the size of the reference dataset, meaning that our results here are not as good as they would be using the full congressional bills dataset. Computers with at least 4GB of memory should be able to run **RTextTools** on medium to large datasets (i.e., up to 30,000 texts) by using the three low-memory algorithms included: general

[1] http://www.congressionalbills.org/research.html

linearized models (Friedman et al., 2010) from the **glmnet** package, maximum entropy (Jurka, 2012) from **maxent**, and support vector machines (Meyer et al., 2012) from **e1071**. For users with larger datasets, we recommend a cloud computing service such as Amazon EC2.

1. Creating a matrix

First, we load our data with `data(USCongress)`, and use the **tm** package (Feinerer et al., 2008) to create a document-term matrix. The `USCongress` dataset comes with **RTextTools** and hence the function `data` is relevant only to data emanating from R packages. Researchers with data in Access, CSV, Excel, or text files will want to load data via the `read_data()` function. Several pre-processing options from the **tm** package are available at this stage, including stripping whitespace, removing sparse terms, word stemming, and stopword removal for several languages.[2] We want readers to be able to reproduce our example, so we set `removeSparseTerms` to .998, which vastly reduces the size of the document-term matrix, although it may also reduce accuracy in real-world applications. Users should consult Feinerer et al. (2008) to take full advantage of preprocessing possibilities. Finally, note that the text column can be encapsulated in a `cbind()` data frame, which allows the user to perform supervised learning on multiple columns if necessary.

```
data(USCongress)

# CREATE THE DOCUMENT-TERM MATRIX
doc_matrix <- create_matrix(USCongress$text, language="english", removeNumbers=TRUE,
                            stemWords=TRUE, removeSparseTerms=.998)
```

2. Creating a container

The matrix is then partitioned into a container, which is essentially a list of objects that will be fed to the machine learning algorithms in the next step. The output is of class `matrix_container` and includes separate train and test sparse matrices, corresponding vectors of train and test codes, and a character vector of term label names. Looking at the example below, `doc_matrix` is the document term matrix created in the previous step, and `USCongress$major` is a vector of document labels taken from the `USCongress` dataset. `trainSize` and `testSize` indicate which documents to put into the training set and test set, respectively. The first 4,000 documents will be used to train the machine learning model, and the last 449 documents will be set aside to test the model. In principle, users do not have to store both documents and labels in a single dataset, although it greatly simplifies the process. So long as the documents correspond to the document labels via the `trainSize` and `testSize` parameters, `create_container()` will work properly. Finally, the `virgin` parameter is set to `FALSE` because we are still in the evaluation stage and not yet ready to classify virgin documents.

```
container <- create_container(doc_matrix, USCongress$major, trainSize=1:4000,
                              testSize=4001:4449, virgin=FALSE)
```

From this point, users pass the container into every subsequent function. An ensemble of up to nine algorithms can be trained and classified.

3. Training models

The `train_model()` function takes each algorithm, one by one, to produce an object passable to `classify_model()`. A convenience `train_models()` function trains all models at once by passing in a vector of model requests.[3] The syntax below demonstrates model creation for all nine algorithms. For expediency, users replicating this analysis may want to use just the three low-memory algorithms: support vector machine (Meyer et al., 2012), glmnet (Friedman et al., 2010), and maximum entropy (Jurka, 2012).[4] The other six algorithms include: scaled linear discriminant analysis (slda) and bagging (Peters and Hothorn, 2012) from **ipred**; boosting (Tuszynski, 2012) from **caTools**; random forest (Liaw and Wiener, 2002) from **randomForest**; neural networks (Venables and Ripley, 2002) from **nnet**; and classification or regression tree (Ripley., 2012) from **tree**. Please see the aforementioned references to

[2] Languages include Danish, Dutch, English, Finnish, French, German, Italian, Norwegian, Portuguese, Russian, Spanish, and Swedish.

[3] `classify_models()` is the corollary to `train_models()`.

[4] Training and classification time may take a few hours for all 9 algorithms, compared to a few minutes for the low memory algorithms.

find out more about the specifics of these algorithms and the packages from which they originate. Finally, additional arguments can be passed to `train_model()` via the `. . .` argument.

```
SVM <- train_model(container,"SVM")
GLMNET <- train_model(container,"GLMNET")
MAXENT <- train_model(container,"MAXENT")
SLDA <- train_model(container,"SLDA")
BOOSTING <- train_model(container,"BOOSTING")
BAGGING <- train_model(container,"BAGGING")
RF <- train_model(container,"RF")
NNET <- train_model(container,"NNET")
TREE <- train_model(container,"TREE")
```

4. Classifying data using trained models

The functions `classify_model()` and `classify_models()` use the same syntax as `train_model()`. Each model created in the previous step is passed on to `classify_model()`, which then returns the classified data.

```
SVM_CLASSIFY <- classify_model(container, SVM)
GLMNET_CLASSIFY <- classify_model(container, GLMNET)
MAXENT_CLASSIFY <- classify_model(container, MAXENT)
SLDA_CLASSIFY <- classify_model(container, SLDA)
BOOSTING_CLASSIFY <- classify_model(container, BOOSTING)
BAGGING_CLASSIFY <- classify_model(container, BAGGING)
RF_CLASSIFY <- classify_model(container, RF)
NNET_CLASSIFY <- classify_model(container, NNET)
TREE_CLASSIFY <- classify_model(container, TREE)
```

5. Analytics

The most crucial step during the machine learning process is interpreting the results, and **RTextTools** provides a function called `create_analytics()` to help users understand the classification of their test set data. The function returns a container with four different summaries: by label (e.g., topic), by algorithm, by document, and an ensemble summary.

Each summary's contents will differ depending on whether the virgin flag was set to `TRUE` or `FALSE` in the `create_container()` function in Step 3. For example, when the virgin flag is set to `FALSE`, indicating that all data in the training and testing sets have corresponding labels, `create_analytics()` will check the results of the learning algorithms against the true values to determine the accuracy of the process. However, if the virgin flag is set to `TRUE`, indicating that the testing set is unclassified data with no known true values, `create_analytics()` will return as much information as possible without comparing each predicted value to its true label.

The label summary provides statistics for each unique label in the classified data (e.g., each topic category). This includes the number of documents that were manually coded with that unique label (`NUM_MANUALLY_CODED`), the number of document sthat were coded using the ensemble method (`NUM_CONSENSUS_CODED`), the number of documents that were coded using the probability method (`NUM_PROBABILITY_CODED`), the rate of over- or under-coding with each method (`PCT_CONSENSUS_CODED` and `PCT_PROBABILITY_CODED`), and the percentage that were correctly coded using either the ensemble method or the probability method (`PCT_CORRECTLY_CODED_CONSENSUS` or `PCT_CORRECTLY_CODED_PROBABILITY`, respectively).

The algorithm summary provides a breakdown of each algorithm's performance for each unique label in the classified data. This includes metrics such as precision, recall, f-scores, and the accuracy of each algorithm's results as compared to the true data.[5]

The document summary provides all the raw data available for each document. By document, it displays each algorithm's prediction (`ALGORITHM_LABEL`), the algorithm's probability score (`ALGORITHM_PROB`), the number of algorithms that agreed on the same label (`CONSENSUS_AGREE`), which algorithm had the highest probability score for its prediction (`PROBABILITY_CODE`), and the original

[5] If a dataset is small (such as the present example), there is a chance that some f-score calculations will report `NaN` for some labels. This happens because not all labels were classified by a given algorithm (e.g., no cases were given a 5). In most real-world datasets, this will not happen.

label of the document (MANUAL_CODE). Finally, the create_analytics() function outputs information pertaining to ensemble analysis, which is discussed in its own section.

Summary and print methods are available for create_analytics(), but users can also store each summary in separate data frames for further analysis or writing to disk. Parameters include the container object and a matrix or cbind() object of classified results.[6]

```
analytics <- create_analytics(container,
                              cbind(SVM_CLASSIFY, SLDA_CLASSIFY,
                                    BOOSTING_CLASSIFY, BAGGING_CLASSIFY,
                                    RF_CLASSIFY, GLMNET_CLASSIFY,
                                    NNET_CLASSIFY, TREE_CLASSIFY,
                                    MAXENT_CLASSIFY))
summary(analytics)

# CREATE THE data.frame SUMMARIES
topic_summary <- analytics@label_summary
alg_summary <- analytics@algorithm_summary
ens_summary <-analytics@ensemble_summary
doc_summary <- analytics@document_summary
```

6. Testing algorithm accuracy

While there are many techniques used to evaluate algorithmic performance (McLaughlin and Herlocker, 2004), the summary call to create_analytics() produces precision, recall and f-scores for analyzing algorithmic performance at the aggregate level. Precision refers to how often a case the algorithm predicts as belonging to a class actually belongs to that class. For example, in the context of the USCongress data, precision tells us what proportion of bills an algorithm deems to be about defense are actually about defense (based on the gold standard of human-assigned labels). In contrast, recall refers to the proportion of bills in a class the algorithm correctly assigns to that class. In other words, what percentage of actual defense bills did the algorithm correctly classify? F-scores produce a weighted average of both precision and recall, where the highest level of performance is equal to 1 and the lowest 0 (Sokolova et al., 2006).

Users can quickly compare algorithm performance. For instance, our working example shows that the SVM, glmnet, maximum entropy, random forest, SLDA, and boosting vastly outperform the other algorithms in terms of f-scores, precision, and recall. For instance, SVM's f-score is 0.65 whereas SLDA's f-score is 0.63, essentially no difference. Thus, if analysts wish to only use one algorithm, any of these top algorithms will produce comparable results. Based on these results, Bagging, Tree, and NNET should not be used singly.[7]

Algorithm	Precision	Recall	F-score
SVM	0.67	0.65	0.65
Glmnet	0.68	0.64	0.64
SLDA	0.64	0.63	0.63
Random Forest	0.68	0.62	0.63
Maxent	0.60	0.62	0.60
Boosting	0.65	0.59	0.59
Bagging	0.52	0.39	0.39
Tree	0.20	0.22	0.18
NNET	0.07	0.12	0.08

Table 1: Overall algorithm precision, recall, and f-scores.

[6] Users who conduct the analysis with the three low-memory algorithms will want to slightly modify the following code.

[7] Note that these results are specific to these data. The overall sample size is somewhat small. Bagging, Tree, and NNET perform well with more data.

7. Ensemble agreement

We recommend ensemble (consensus) agreement to enhance labeling accuracy. Ensemble agreement simply refers to whether multiple algorithms make the same prediction concerning the class of an event (i.e., did SVM and maximum entropy assign the same label to the text?). Using a four-ensemble agreement approach, Collingwood and Wilkerson (2012) found that when four of their algorithms agree on the label of a textual document, the machine label matches the human label over 90% of the time. The rate is just 45% when only two algorithms agree on the text label.

RTextTools includes `create_ensembleSummary()`, which calculates both recall accuracy and coverage for **n** ensemble agreement. Coverage simply refers to the percentage of documents that meet the recall accuracy threshold. For instance, say we find that when seven algorithms agree on the label of a bill, our overall accuracy is 90% (when checked against our true values). Then, let's say, we find that only 20% of our bills meet that criterion. If we have 10 bills and only two bills meet the seven ensemble agreement threshold, then our coverage is 20%. Mathematically, if k represents the percent of cases that meet the ensemble threshold, and n represents total cases, coverage is calculated in the following way:

$$\text{Coverage} = \frac{k}{n} \tag{1}$$

Users can test their accuracy using a variety of ensemble cut-points, which is demonstrated below. Table 2 reports the coverage and recall accuracy for different levels of ensemble agreement. The general trend is for coverage to decrease while recall increases. For example, just 11% of the congressional bills in our data have nine algorithms that agree. However, recall accuracy is 100% for those bills when the 9 algorithms do agree. Considering that 90% is often social scientists' inter-coder reliability standard, one may be comfortable using a 6 ensemble agreement with these data because we label 66% of the data with accuracy at 90%.[8]

```
create_ensembleSummary(analytics@document_summary)
```

	Coverage	Recall
n >= 2	1.00	0.77
n >= 3	0.98	0.78
n >= 4	0.90	0.82
n >= 5	0.78	0.87
n >= 6	0.66	0.90
n >= 7	0.45	0.94
n >= 8	0.29	0.96
n >= 9	0.11	1.00

Table 2: Ensemble agreement coverage and recall.

8. Cross validation

Users may use n-fold cross validation to calculate the accuracy of each algorithm on their dataset and determine which algorithms to use in their ensemble. **RTextTools** provides a convenient `cross_validate()` function to perform n-fold cross validation. Note that when deciding the appropriate n-fold it is important to consider the total sample size so that enough data are in both the train and test sets to produce useful results.

```
SVM <- cross_validate(container, 4, "SVM")
GLMNET <- cross_validate(container, 4, "GLMNET")
MAXENT <- cross_validate(container, 4, "MAXENT")
SLDA <- cross_validate(container, 4, "SLDA")
BAGGING <- cross_validate(container, 4, "BAGGING")
BOOSTING <- cross_validate(container, 4, "BOOSTING")
```

[8] This result is excellent, given that the training data consist of just a few thousand observations.

```
RF <- cross_validate(container, 4, "RF")
NNET <- cross_validate(container, 4, "NNET")
TREE <- cross_validate(container, 4, "TREE")
```

9. Exporting data

Finally, some users may want to write out the newly labeled data for continued manual labeling on texts that do not achieve high enough accuracy requirements during ensemble agreement. Researchers can then have human coders verify and label these data to improve accuracy. In this case, the document summary object generated from `create_analytics` can be easily exported to a CSV file.

```
write.csv(analytics@document_summary, "DocumentSummary.csv")
```

Conclusion

Although a user can get started with **RTextTools** in less than ten steps, many more options are available that help to remove noise, refine trained models, and ultimately improve accuracy. If you want more control over your data, please refer to the documentation bundled with the package. Moreover, we hope that the package will become increasingly useful and flexible over time as advanced users develop add-on functions. **RTextTools** is completely open-source, and a link to the source code repository as well as a host of other information can be found at the project's website – http://www.rtexttools.com.

Bibliography

E. S. Adler and J. Wilkerson. *Congressional Bills Project*. URL http://congressionalbills.org/, 2004. [p6]

F. Baumgartner, S. De Boef, and A. Boydstun. *The Decline of the Death Penalty and the Discovery of Innocence*. Cambridge University Press, 2008. [p6]

L. Collingwood and J. Wilkerson. Tradeoffs in accuracy and efficiency in supervised learning methods. *Journal of Information Technology & Politics*, 9(3):298–318, 2012. [p10]

I. Feinerer, K. Hornik, and D. Meyer. Text mining infrastructure in R. *Journal of Statistical Software*, 25 (5):1–54, 2008. [p7]

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010. [p7]

B. Jones, H. Larsen-Price, and J. Wilkerson. Representation and American governing institutions. *The Journal of Politics*, 71(01):277–290, 2009. [p6]

T. P. Jurka. maxent: An R package for low-memory multinomial logistic regression with support for semi-automated text classification. *The R Journal*, 4(1):56–59, June 2012. [p7]

A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2(3):18–22, 2002. [p7]

M. McLaughlin and J. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–336. ACM, 2004. [p9]

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2012. URL http://CRAN.R-project.org/package=e1071. R package version 1.6-1. [p7]

A. Peters and T. Hothorn. *ipred: Improved Predictors*, 2012. URL http://CRAN.R-project.org/package=ipred. R package version 0.8-13. [p7]

B. Ripley. *tree: Classification and Regression Trees*, 2012. URL http://CRAN.R-project.org/package=tree. R package version 1.0-31. [p7]

M. Sokolova, N. Japkowicz, and S. Szpakowicz. Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In *AI 2006: Advances in Artificial Intelligence*, pages 1015–1021, Springer, 2006. [p9]

J. Tuszynski. *caTools: Tools: Moving Window Statistics, GIF, Base64, ROC AUC, etc.*, 2012. URL http://CRAN.R-project.org/package=caTools. R package version 1.13. [p7]

W. Venables and B. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. [p7]

Timothy P. Jurka
Department of Political Science
University of California, Davis
One Shields Avenue
Davis, CA 95616
USA
tpjurka@ucdavis.edu

Loren Collingwood
Department of Political Science
University of California, Riverside
900 University Avenue
Riverside, CA 92521
USA
loren.collingwood@ucr.edu

Amber E. Boydstun
Department of Political Science
University of California, Davis
One Shields Avenue
Davis, CA 95616
USA
aboydstun@ucdavis.edu

Emiliano Grossman
Sciences Po /CEE
28, rue des Saints-Pères
75007 Paris
France
emiliano.grossman@sciences-po.fr

Wouter van Atteveldt
Communication Science Department
Vrije Universiteit
de Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
wouter@vanatteveldt.com

Generalized Simulated Annealing for Global Optimization: The GenSA Package

An Application to Non-Convex Optimization in Finance and Physics

by Yang Xiang, Sylvain Gubian, Brian Suomela and Julia Hoeng

Abstract Many problems in statistics, finance, biology, pharmacology, physics, mathematics, economics, and chemistry involve determination of the global minimum of multidimensional functions. R packages for different stochastic methods such as genetic algorithms and differential evolution have been developed and successfully used in the R community. Based on Tsallis statistics, the R package GenSA was developed for generalized simulated annealing to process complicated non-linear objective functions with a large number of local minima. In this paper we provide a brief introduction to the R package and demonstrate its utility by solving a non-convex portfolio optimization problem in finance and the Thomson problem in physics. **GenSA** is useful and can serve as a complementary tool to, rather than a replacement for, other widely used R packages for optimization.

Introduction

Many problems in statistics, biology, physics, mathematics, economics, and chemistry involve the determination of the global minimum of multidimensional functions (Agostini et al., 2006; Serra et al., 1997; Guire et al., 2010; Xiang et al., 1997; Xiang and Gong, 2000a; Xiang et al., 2000). Methods of optimization can generally be classified as either deterministic or stochastic. For simple non-convex functions with only a few dimensions and well separated local minima, standard deterministic methods such as simplex optimization, steepest descent method, and the quasi-Newton method can provide reasonable results. While being fast, deterministic methods have the tendency to trap the system within a local minimum. By comparison, many stochastic methods are far less likely to get stuck in a local minimum, and may find a good approximation of the global minimum using a modest amount of computation. Many stochastic methods have been developed for the solution of global optimization problems such as genetic algorithms (Holland, 1975), evolution algorithms (Storn and Price, 1997), simulated annealing (Kirkpatrick et al., 1983), and taboo search (Glover et al., 1993; Cvijovic and Klinowski, 2002, 1995).

In metallurgy, annealing a molten metal causes it to reach its crystalline state which is the global minimum in terms of thermodynamic energy. The simulated annealing algorithm was developed to simulate the annealing process to find a global minimum of the objective function (Kirkpatrick et al., 1983). In the simulated annealing algorithm, the objective function is treated as the energy function of a molten metal and one or more artificial temperatures are introduced and gradually cooled, analagous to the annealing technique. This artificial temperature (or set of temperatures) acts as a source of stochasticity, which is convenient for the systems to eventually escape from local minima. Near the end of the annealing process, the system is hopefully inside the attractive basin of the global minimum (or in one of the global minima if more than one global minimum exists).

In contrast to the simulation of the annealing process of molten metal, genetic algorithms (Holland, 1975) were developed by mimicing the process of natural evolution. A population of strings which encode candidate solutions for an optimization problem evolve over many iterations toward better solutions. In general the solutions are represented by bitstrings, but other encodings such as floating-point numbers are also widely used. The evolution usually starts from a population of randomly generated individuals. In each generation, the fitness of each individual in the population is evaluated. New members of the population in the next generation are generated by cross-over, mutation, and selection (based on their fitness). Differential evolution belongs to a class of genetic algorithms.

The basic idea behind the taboo search method (Glover et al., 1993) is to forbid the search to return to points already visited in the (usually discrete) search space, at least for the upcoming few steps. Similar to simulated annealing, taboo search can temporarily accept new solutions which are worse than earlier solutions, in order to avoid paths already investigated. Taboo search has traditionally been applied to combinatorial optimization problems and it has been extended to be applicable to continuous global optimization problems by a discrete approximation (encoding) of the problem (Cvijovic and Klinowski, 2002, 1995).

For a review of stochastic optimization algorithms, please refer to Fouskakis and Draper (2002).

The R language and environment for statistical computing enables rapid prototyping of algorithms, access to a wide range of tools for statistical modeling, and the ability to easily generate customized plots of results. These advantages make use of R preferable in many situations over the use of other programming languages like Java, C++, Fortran, and Pascal (Mullen et al., 2011).

Theussl (2011) provided a comprehensive summary of the currently available R packages for solving optimization problems. No one method is the best for all the optimization problems. The best method for one specific problem depends on the problem itself. For example, simulated annealing could be chosen for optimization problems such as scheduling in the multiprocessor flowshop (Figielska, 2009), or pathway-based microarray analysis (Pavlidis et al., 2011), while genetic algorithms could be more appropriate for optimization problems such as design of an ATM network (Thompson and Bilbro, 2000). R users can choose the most appropriate method/package to handle different types of global optimization problems facilitated by the work of Mullen et al. (2011); Ardia et al. (2011); Mebane Jr and Sekhon (2011); Powell (2009) and other contributors listed in Theussl (2011). For example, the Nelder-Mead and BFGS quasi newton method in the function `optim` are appropriate to handle non-smooth and smooth functions with few local minima; packages for stochastic searches, such as **DEoptim** (Mullen et al., 2011; Ardia et al., 2011) and **rgenoud** (Mebane Jr and Sekhon, 2011), are a good choice for more complicated objective functions with many local minima. While the simulated annealing algorithm is also suitable to solve complicated objective functions with many local minima, the only package of simulated annealing serving as a general purpose continuous solver in R is `sann` in `optim` (Theussl, 2011). Several R packages, **ConsPlan** (VanDerWal and Januchowski, 2010), **likelihood** (Murphy, 2008), **dclone** (Sólymos, 2010), and **subselect** (Cerdeira et al., 2011), make use of the simulated annealing algorithm within their specific applications. As `sann` in `optim` cannot handle simple box constraints and it performs poorly compared to **DEoptim** for the Rastrigin function (Ardia et al., 2011), many R users would benefit from a new R package for simulated annealing which is suitable for box constraints and quickly solves global optimization problems.

Here we have developed an innovative R package **GenSA**, which is an implementation of modified generalized simulated annealing, which outperforms the classical simulated annealing algorithm (Xiang et al., 1997; Xiang and Gong, 2000a,b; Serra et al., 1997). **GenSA** can process complicated non-linear objective functions with a large number of local minima. The kernel function of **GenSA** is written in C++ to ensure the package runs as efficiently as possible.

In the remainder of this paper we will elaborate further on the background and improvements of GSA (in Section 1) and on the content of the package **GenSA** (in Section 2). The utility of this R package for financial and physical applications is demonstrated by solving a non-convex portfolio optimization problem and the famous Thomson problem in physics, respectively (in Section 3).

Generalized simulated annealing

Classical simulated annealing (CSA) was proposed by Kirkpatrick et al. (1983). Due to the inherent statistical nature of simulated annealing, in principle local minima can be hopped over more easily than for gradient methods. Using the simulated annealing technique, one or more artificial temperatures are introduced and gradually cooled to simulate thermal noise. A fast simulated annealing (FSA) method was proposed by Szu and Hartley (Szu and Hartley, 1987). CSA and FSA can be generalized according to Tsallis statistics with a unified picture called the generalized simulated annealing algorithm (Tsallis and Stariolo, 1996). GSA uses a distorted Cauchy-Lorentz visiting distribution, with its shape controlled by the parameter q_v

$$g_{q_v}(\Delta x(t)) \propto \frac{\left[T_{q_v}(t)\right]^{-\frac{D}{3-q_v}}}{\left[1+(q_v-1)\frac{(\Delta x(t))^2}{\left[T_{q_v}(t)\right]^{\frac{2}{3-q_v}}}\right]^{\frac{1}{q_v-1}+\frac{D-1}{2}}}. \tag{1}$$

Here t is the artificial time. This visiting distribution is used to generated a trial jump distance $\Delta x(t)$ of variable $x(t)$ under artificial temperature $T_{q_v}(t)$. The trial jump is accepted if it is downhill (in terms of the objective function). If the jump is uphill it might be accepted according to an acceptance probability. A generalized Metropolis algorithm is used for the acceptance probability:

$$p_{q_a} = \min\left\{1, \left[1-(1-q_a)\beta\Delta E\right]^{\frac{1}{1-q_a}}\right\}, \tag{2}$$

where q_a is a parameter. For $q_a < 1$, zero acceptance probability is assigned to the cases where

$$\left[1-(1-q_a)\beta\Delta E\right] < 0. \tag{3}$$

The artificial temperature $T_{q_v}(t)$ is decreased according to

$$T_{q_v}(t) = T_{q_v}(1)\frac{2^{q_v-1}-1}{(1+t)^{q_v-1}-1}. \quad (4)$$

When $q_v = 1$ and $q_a = 1$, GSA recovers CSA; when $q_v = 2$ and $q_a = 1$, GSA recovers FSA. When $q_v > 2$, the cooling is faster than that of CSA and FSA. When $T \to 0$, GSA behaves like the steepest descent algorithm. GSA not only converges faster than FSA and CSA, but also has the ability to escape from a local minimum more easily than FSA and CSA. Since GSA has a large probability to have a long jump, the probability of finding the global minimum with GSA is larger than that with FSA and CSA. Bigger q_v (< 3) makes the cooling faster and jumping further. Negative q_a with bigger absolute value makes GSA accept less hill climbing. Default values of q_v and q_a in our package **GenSA** are set to be 2.62 and -5 respectively according to our experience and our package works well with the default values in many applications. We also provide users the flexibility of changing the value of q_v and q_a easily in the *control* argument of **GenSA** for advanced usage of **GenSA**. For example, users can set any value between 2 and 3 for q_v and any value < 0 for q_a.

A simple technique to speed up the convergence is to set the acceptance temperature equal to the visiting temperature divided by the number of time steps, i.e. $T_{q_a} = \frac{T_{q_v}}{t}$. Our testing shows that this simple technique works well (Xiang et al., 2000), so this technique is used in **GenSA** to speed up the convergence.

For more details on GSA, we refer the readers to Tsallis and Stariolo (1996), Xiang and Gong (2000a) and Xiang et al. (1997).

The package GenSA

The core function of package **GenSA** is the function `GenSA`, whose arguments are:

- `par`: Numeric vector. Initial values for the parameters to be optimized over. Default is `NULL`, in which case initial values will be generated automatically.
- `lower`: Numeric vector with length of `par`. Lower bounds for the parameters to be optimized over.
- `upper`: Numeric vector with length of `par`. Upper bounds for the parameters to be optimized over.
- `fn`: A function to be minimized, with first argument the vector of parameters over which minimization is to take place. It should return a scalar result.
- `...`: allows the user to pass additional arguments to the function `fn`.
- `control`: A list of control parameters, discussed below.

The `control` argument is a list that can be used to control the behavior of the algorithm. Some components of `control` are the following:

- `maxit`: Integer. Maximum number of iterations of the algorithm. Default value is 5000, which could be increased by the user for the optimization of a very complicated objective function.
- `threshold.stop`: Numeric. The program will stop when the objective function value is <= `threshold.stop`. Default value is NULL.
- `smooth`: Logical. TRUE when the objective function is smooth, or differentiable almost everywhere, and FALSE otherwise. Default value is TRUE.
- `max.call`: Integer. Maximum number of calls of the objective function. Default is 10000000.
- `max.time`: Numeric. Maximum running time in seconds.
- `temperature`: Numeric. Initial value for temperature.

The default values of the control components are set for a complicated optimization problem. For typical optimization problems with medium complexity, `GenSA` can find a reasonable solution quickly so the user is strongly recommended to let `GenSA` stop earlier by setting `threshold.stop` to the expected function value, or by setting `max.time` if the user just want to run `GenSA` for `max.time` seconds, or by setting `max.call` if the user just want to run `GenSA` within `max.call` function calls. Please refer to the examples below. For very complicated optimization problems, the user is recommended to increase `maxit` and `temperature`.

The returned value is a list with the following fields:

- `par`: The best set of parameters found.
- `value`: The value of `fn` corresponding to `par`.
- `trace.mat`: A matrix which contains the history of the algorithm. (By columns: Step number, temperature, current objective function value, current minimum objective function value).
- `counts`: Total number of calls of the objective function.

For more details about **GenSA**, the reader can refer to the manual (by typing `?GenSA`).

Packages **DEoptim** and **rgenoud** have been widely used to successfully solve optimization problems arising in diverse fields (Ardia et al., 2011; Mullen et al., 2011; Mebane Jr and Sekhon, 2011). Since these two packages both use evolutionary strategies (Ardia et al., 2011), we have chosen one of them, **DEoptim**, as our gold standard for comparison purposes of various global optimization packages. Similar to Ardia et al. (2011), let us briefly illustrate the usage of package **GenSA** with the minimization of the Rastrigin function with 2 dimensions, which is a standard test function for global optimization:

```
> Rastrigin <- function(x) {
+   fn.call <<- fn.call + 1
+   sum(x^2 - 10 * cos(2 * pi * x)) + 10 * length(x)
+ }
```

The Rastrigin function with 2 dimensions has many local minima. The global minimum $f^{opt} = 0$ at point $x^{opt} = (0, 0)$. Please note that the dimension of `x` of the above `Rastrigin` function can be more than 2.

Since we chose **DEoptim** as our gold standard for global optimization packages, let's run **DEoptim** first:

```
> options(digits = 10)
> dimension <- 2
> lower <- rep(-5.12, dimension)
> upper <- rep(5.12, dimension)
> set.seed(1234)
> sink("tmp.txt")
> fn.call <<- 0
> library(DEoptim)
> out.DEoptim <- DEoptim(fn = Rastrigin, lower = lower, upper = upper,
+                        control = list(storepopfrom = 1))
> sink(NULL)
> out.DEoptim$optim[c(1, 2, 3)]
$bestmem
            par1             par2
-4.775211422e-07  6.390004611e-08

$bestval
[1] 4.60502747e-11

$nfeval
[1] 402

> cat("DEoptim call functions", fn.call, "times.\n")
DEoptim call functions 4020 times.
```

The best value that **DEoptim** found is 4.60502747e-11, which is fairly close to the global minimum 0. However the latest version of **DEoptim**, 2.2.1, gave a wrong number of function calls `nfeval`, 402, which should be 4020 as shown by `fn.call`.

The simulated annealing solver `sann` in `optim` is applied to the Rastrigin function.

```
> set.seed(1234)
> x.ini <- lower + runif(length(lower)) * (upper - lower)
> fn.call <- 0
> out.sann <- optim(par = x.ini, fn = Rastrigin, method = "SANN")
> out.sann[c("value","par","counts")]
$value
[1] 3.980605068

$par
```

```
[1] -1.98836973902 -0.00123560529

$counts
function gradient
   10000       NA
```

sann does not find the global minimum although 10000 function calls (the default stopping criterion in sann) are performed.

GenSA also searches for a minimum of the Rastrigin function between the same lower and upper bounds. A call to **GenSA** can be made as follows:

```
> set.seed(1234)
> library(GenSA)
> expected.val <- 0
> absTol <- 1e-13
> fn.call <- 0
> out.GenSA <- GenSA(par = NULL, lower = lower, upper = upper, fn = Rastrigin,
+                    control = list(threshold.stop = expected.val + absTol))
> out.GenSA[c("value", "par", "counts")]
$value
[1] 0

$par
[1]  2.750668687e-12 -2.889218652e-12

$counts
[1] 196

> cat("GenSA call functions", fn.call, "times.\n")
GenSA call functions 196 times.
```

The above call specifies a random vector as initial values for the parameters by setting par as NULL, the lower and upper bounds on the parameters, and, via the control argument, that we will stop searching after **GenSA** finds the expected global minimum $f^{opt} = 0$ within the absolute tolerance $\Delta = 1e{-}13$. **GenSA** actually found the global minimum $f^{opt} = 0$ after 196 function calls. **GenSA** and **DEoptim** both converge to the the global minimum but **GenSA** obtains a more accurate result with fewer function calls than **DEoptim**. sann performs poorly since it does not find the global minimum after a much larger number of function calls (10000) compared to **DEoptim** and **GenSA**.

The result of **DEoptim** can be further improved with a local search, large-scale bound-constrained BFGS (Byrd et al., 1995; Zhu et al., 1997), as below:

```
> out.DEoptim_BFGS <- optim(par = out.DEoptim$optim$bestmem, fn = Rastrigin,
+                           lower = lower, upper = upper, method = "L-BFGS-B")
> out.DEoptim_BFGS[c("value","par")]
$value
[1] 0

$par
            par1             par2
-9.362433236e-12  1.250185258e-12
```

So **DEoptim** + large-scale bound-constrained BFGS generated a comparable result to **GenSA**, however, with more function calls. L-BFGS-B and bobyqa are good local search methods for smooth functions while Nelder-Mead is recommended for local search of non-smooth functions although Nelder-Mead can manage both smooth and non-smooth functions (Zhu et al., 1997; Nash, 1990; Powell, 2009). It could be a good idea to run a local search for further refinement after running **DEoptim**. The user of **GenSA** does not need to run a local search for refinement after running **GenSA** since the local search refinement was performed within **GenSA**.

The Rastrigin function with 30 dimentions belongs to the most difficult class of problems for many optimization algorithms (including evolutionary programming) (Yao et al., 1999; Mebane Jr and Sekhon, 2011). However **GenSA** can find the global minimum in no more than 2 seconds (CPU: Xeon E5450), as shown below.

```
> set.seed(1234)
> dimension <- 30
> lower <- rep(-5.12, dimension)
```

```
> upper <- rep(5.12, dimension)
> fn.call <- 0
> out.GenSA <- GenSA(lower = lower, upper = upper, fn = Rastrigin,
+                    control = list(max.time=1.9, verbose=TRUE))
> out.GenSA[c("value")]
$value
[1] 0
```

Of course the time for solving the above problem using **GenSA** may vary depending on the computing power used. **DEoptim** and sann did not find the global minimum of Rastrigin function with 30 dimensions by using default settings.

To compare the performance of the above packages/methods more precisely, **GenSA**, **DEoptim**, sann, and **DEoptim**+ large-scale bound-constrained BFGS (denoted as DEoptim_BFGS) were run 100 times randomly. The testing function was Rastrigin with 2 dimensions. We noticed that the implementation in the package **DEoptim** is very flexible, but only default values of parameters were used in the comparison. Similarly only default values of parameters in **GenSA** were used in the comparison. Various absolute tolerances $\Delta \in \{1e{-}05,\ 1e{-}07,\ 1e{-}08,\ 1e{-}09\}$ were explored. The global minimum $f^{opt} = 0$ was considered to have been reached if the current function value f surpasses $f^{opt} + \Delta$. A run was successful if the global minimum was reached. Figures 1 and 2 show the percentage of the 100 runs that were successful, and the average number of function calls to reach the global minimum (for the successful runs), respectively. The error bar in Figure 2 represents the average number of function calls $\pm$ its standard error (SE).

None of the 100 runs for sann were successful, so the result of sann is not shown in Figure 1 and Figure 2.

In Figure 1, the percentage of successful runs for **DEoptim** was 88% for $\Delta = 1e{-}05$, but decreased to 22% for $\Delta = 1e{-}09$. The percentage of successful runs for **GenSA** and **DEoptim_BFGS** were 100% and 98% respectively for all absolute tolerances Δ. **GenSA**, **DEoptim** and **DEoptim_BFGS** had a similar percentage of successful runs when the absolute tolerance Δ was large, while **GenSA** and **DEoptim_BFGS** were much more likely to be successful than **DEoptim** as the absolute tolerance Δ was set to smaller values.

In Figure 2, the average number of function calls for **GenSA** was 479 for $\Delta = 1e{-}05$, and increased only slightly to 483 for $\Delta = 1e{-}09$. The average number of function calls for **DEoptim** was 2692 for $\Delta = 1e{-}05$, and increased significantly to 3344 for $\Delta = 1e{-}09$. The average number of function calls for **DEoptim_BFGS** was 2829 for $\Delta = 1e{-}05$, and increased to 3877 for $\Delta = 1e{-}09$. A student t-test was performed to determine if the average number of function calls of **GenSA** was significantly smaller than the average number of function calls of **DEoptim** and **DEoptim_BFGS**. The p-values shown above the error bars in Figure 2 indicate that the average number of function calls of **GenSA** is significantly smaller than the average number of function calls of both **DEoptim** and **DEoptim_BFGS**. The Wilcoxon rank-sum test gave a similar result.

GenSA, **DEoptim**, **DEoptim_BFGS** solve the Rastrigin problem more efficiently than sann. To show how the performance of the above R packages change with the dimensions of testing functions, we test them in more functions: generalized Rosenbrock function (ROS) (Mebane Jr and Sekhon, 2011) with 4 dimensions (dimension = 2, 10, 20, 30), Rastrigin function (RAS) (Mebane Jr and Sekhon, 2011) with 4 dimensions (dimension = 2, 10, 20, 30), Branin function (BRA) (Serra et al., 1997), Goldstein-Prices function (GP) (Serra et al., 1997). The range of coordindates of functions Rosenbrock, Rastrigin, and Goldstein-Prices, are $[-30,\ 30]$, $[-5.12,\ 5.12]$, and $[-2,\ 2]$, respectively. The range of the first and the second coordinate of function Branin are $[-5,\ 10]$ and $[0,\ 15]$ respectively. Four different tolerance levels, $1e{-}5,\ 1e{-}7,\ 1e{-}8,\ 1e{-}9$, were used. The results for tolerance level $1e{-}08$ are shown in Table 1. For Rosenbrock function with low dimension 2, **GenSA**, **DEoptim** and **DEoptim_BFGS** found the global minimum with 100% success. For Rosenbrock function with dimension $=\ 10,\ 20,\ 30$, **DEoptim** and **DEoptim_BFGS** failed to find the global minimum (tolerance level $1e{-}08$), while **GenSA** found the global minimum (tolerance level $1e{-}08$) with a 100% success rate. For Rosenbrock function with dimension $2,\ 10,\ 20,\ 30$, the average number of function calls of **GenSA** is significantly smaller than that of **DEoptim** and **DEoptim_BFGS**. For Rastrigin function with dimension $\leq\ 30$, the success rate of **GenSA** is 100% regardless of the number of dimensions, but the success rate of **DEoptim** and **DEoptim_BFGS** decreased drastically with an increasing number of dimensions. **DEoptim** failed to find the global minimum (tolerance level $1e{-}08$) of the Rastrigin function for dimension ≥ 20. For Branin function and Goldstein-Prices function, although **GenSA**, **DEoptim** and **DEoptim_BFGS** found the global minimum with a 100% success rate, the average number of function calls of **GenSA** is significantly smaller than that of **DEoptim** and **DEoptim_BFGS**.

Please note that the performance testing was based on the usage of the R packages with only the default parameter settings of **GenSA**, **DEoptim**, and optim.

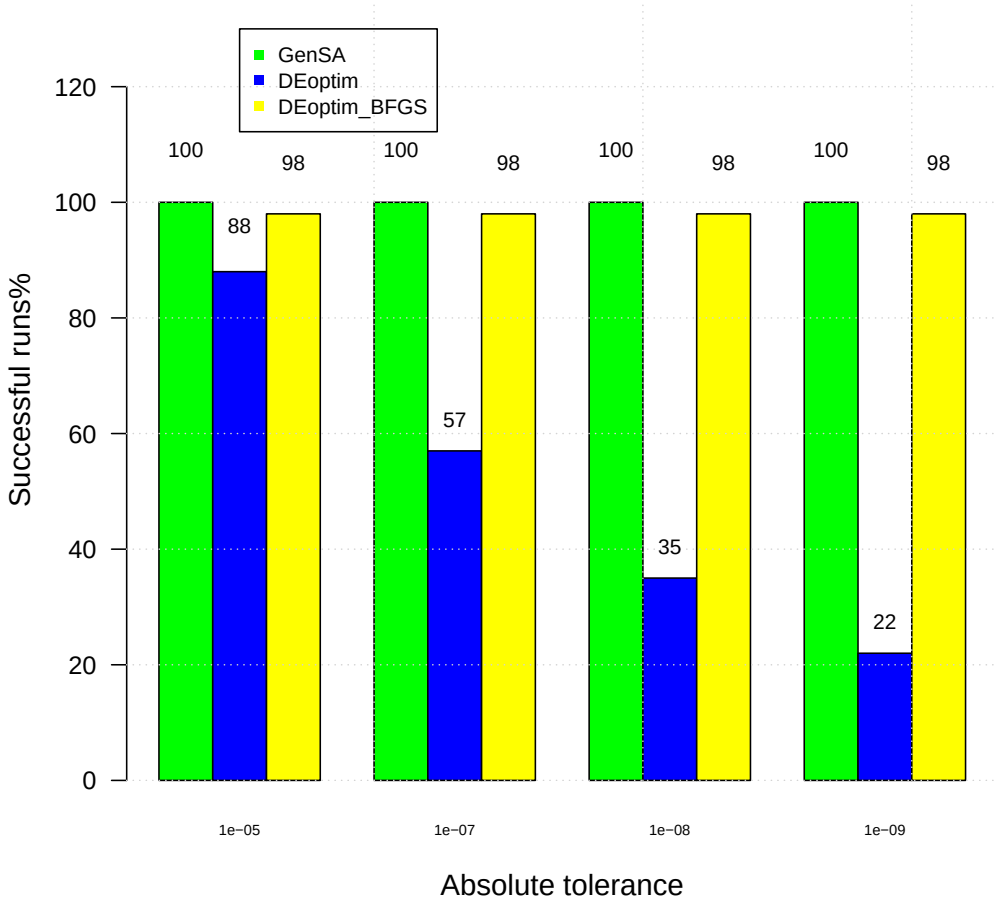


Figure 1: Percentage of successful runs vs. different absolute tolerances for **GenSA** (green), **DEoptim** (blue), and **DEoptim_BFGS** (yellow) in the test function Rastrigin. None of the runs for sann was successful, so the result of sann is not shown in this figure.

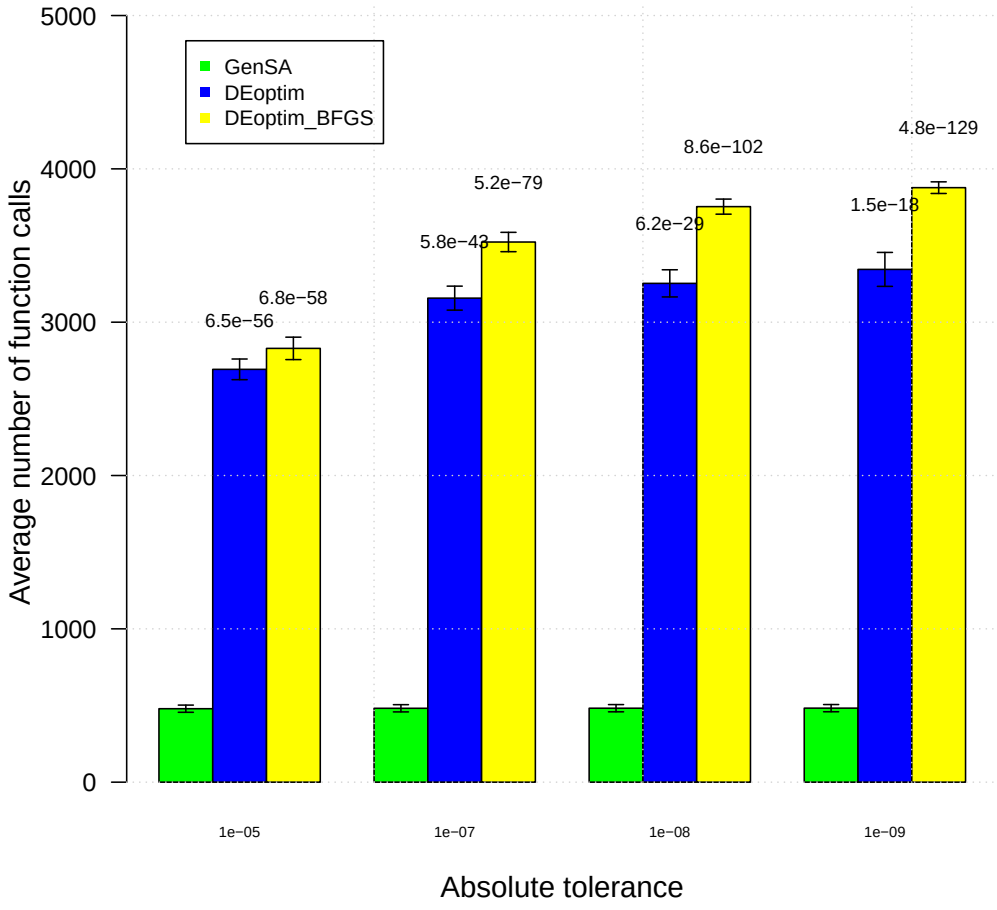


Figure 2: Average number of function calls to reach the global minimum in the successful runs for various absolute tolerances using **GenSA** (green), **DEoptim** (blue), and **DEoptim_BFGS** (yellow) in the test function Rastrigin. The error bar represents the average number of function calls $\pm$ its standard error. The p-values for comparison **DEoptim** and **DEoptim_BFGS** vs. **GenSA** are found above the error bar. None of the runs for sann was successful, so the result of sann is not shown in this figure.

func	GenSA	DEoptim	DEoptim-BFGS	func	GenSA	DEoptim	DEoptim-BFGS
ROS-2D	100.0% 106.0 (B) 1617.8 (A) ± 167.9 (S) 4689.0 (W) 1618.7 (T)	100.0% 1561.0 (B) 2483.1 (A) ± 50.5 (S) 3864.0 (W) 4020.0 (T)	100.0% 1561.0 (B) 2483.1 (A) ± 50.5 (S) 3864.0 (W) 4108.9 (T)	**RAS-10D**	100.0% 2919.0 (B) 5878.2 (A) ± 165.8 (S) 11820.0 (W) 5879.4 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 20100.0 (T)	2.0% 20248.0 (B) 20258.5 (A) ± 10.5 (S) 20269.0 (W) 20290.3 (T)
ROS-10D	100.0% 5545.0 (B) 17562.3 (A) ± 696.3 (S) 31234.0 (W) 17564.1 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 20100.0 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 21657.8 (T)	**RAS-20D**	100.0% 6869.0 (B) 14682.6 (A) ± 318.2 (S) 21893.0 (W) 14683.6 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 40200.0 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 40585.8 (T)
ROS-20D	100.0% 11866.0 (B) 33547.9 (A) ± 1401.9 (S) 65427.0 (W) 33552.2 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 40200.0 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 45165.9 (T)	**RAS-30D**	100.0% 12713.0 (B) 27820.7 (A) ± 683.2 (S) 56432.0 (W) 27823.6 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 60300.0 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 60922.8 (T)
ROS-30D	100.0% 30192.0 (B) 52874.3 (A) ± 1403.8 (S) 117795.0 (W) 52878.8 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 60300.0 (T)	0.0% NA (B) NA (A) ± NA (S) NA (W) 67613.9 (T)	**BRA**	100.0% 26.0 (B) 35.7 (A) ± 0.7 (S) 51.0 (W) 36.7 (T)	100.0% 287.0 (B) 788.9 (A) ± 27.4 (S) 1769.0 (W) 4020.0 (T)	100.0% 287.0 (B) 788.9 (A) ± 27.4 (S) 1769.0 (W) 4046.3 (T)
RAS-2D	100.0% 41.0 (B) 482.4 (A) ± 23.6 (S) 1242.0 (W) 483.5 (T)	35.0% 1975.0 (B) 3253.3 (A) ± 88.6 (S) 3991.0 (W) 4020.0 (T)	98.0% 1975.0 (B) 3753.6 (A) ± 49.2 (S) 4041.0 (W) 4115.5 (T)	**GP**	100.0% 41.0 (B) 158.7 (A) ± 11.5 (S) 585.0 (W) 159.7 (T)	100.0% 794.0 (B) 1023.0 (A) ± 9.4 (S) 1263.0 (W) 4020.0 (T)	100.0% 794.0 (B) 1023.0 (A) ± 9.4 (S) 1263.0 (W) 4125.0 (T)

Table 1: Comparison result for tolerance level 1e-08. ROS: Rosenbrock function; RAS: Rastrigin function ; BRA: Branin function; GP: Goldstein-Price function. For ROS and RAS, the number after the hyphen gives the dimension. E.g. ROS-30D is Rosenbrock with 30 dimensions. For each cell in this table, we show the percentage of successful runs %, minimum (B), average (A), standard error (S), and maximum (W), of number of function calls to reach the global minimum among all the successful runs, and average number of total function calls among the 100 runs (denoted by T) respectively.

Derivative-based deterministic methods, e.g. BFGS, are recommended for smooth functions with only a few local minima such as generalized Rosenbrock functions with few dimensions. Pure stochastic methods such as differential evolution and generalized simulated annealing are for complicated functions with multiple minima such as the Rastrigin function, since local minima can be hopped much more easily in stochastic methods than in deterministic methods, due to the inherent statistical nature of stochastic methods.

Application

Risk allocation portfolios

Mean-risk models were developed in the 1950s for portfolio selection problems. Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR) are the most popular measures of downside risk. Mullen et al. (2011) and Ardia et al. (2011) used **DEoptim** to find the portfolio weights for which the portfolio has the lowest CVaR and each investment can contribute at most 22.5% to total portfolio CVaR risk. For details, please refer to Mullen et al. (2011); Ardia et al. (2011). The code for objective function in portfolio optimization are from Ardia et al. (2011).

```
> library("quantmod")
> tickers <- c("GE", "IBM", "JPM", "MSFT", "WMT")
> getSymbols(tickers, from = "2000-12-01", to = "2010-12-31")
[1] "GE"   "IBM"  "JPM"  "MSFT" "WMT"
> P <- NULL
> for(ticker in tickers) {
+   tmp <- Cl(to.monthly(eval(parse(text = ticker))))
+   P <- cbind(P, tmp)
+ }
> colnames(P) <- tickers
> R <- diff(log(P))
> R <- R[-1,]
```

```
> mu <- colMeans(R)
> sigma <- cov(R)
> library("PerformanceAnalytics")
> pContribCVaR <- ES(weights = rep(0.2, 5),
+                    method = "gaussian", portfolio_method = "component",
+                    mu = mu, sigma = sigma)$pct_contrib_ES
> obj <- function(w) {
+   fn.call <<- fn.call + 1
+   if (sum(w) == 0) { w <- w + 1e-2 }
+   w <- w / sum(w)
+   CVaR <- ES(weights = w,
+              method = "gaussian", portfolio_method = "component",
+              mu = mu, sigma = sigma)
+   tmp1 <- CVaR$ES
+   tmp2 <- max(CVaR$pct_contrib_ES - 0.225, 0)
+   out <- tmp1 + 1e3 * tmp2
+   return(out)
+ }
```

We first run **DEoptim** with the same setting as in Ardia et al. (2011).

```
> set.seed(1234)
> fn.call <- 0
> sink("tmp.txt")
> out.DEoptim <- DEoptim(fn = obj, lower = rep(0, 5), upper = rep(1, 5))
> sink(NULL)
> fn.call.DEoptim <- fn.call
> out.DEoptim$optim$bestval
[1] 0.1142884416
> out.DEoptim$optim$nfeval
[1] 402
> cat("DEoptim call functions", fn.call.DEoptim, "times.\n")
DEoptim call functions 10050 times.
```

The minimum found by **DEoptim** was 0.1142884416 after 10050 function calls. The latest version of **DEoptim**, 2.2.1, gave a wrong number of function calls `nfeval` again. Nelder-Mead method can be used for further refinement since this objective function is non-smooth.

```
> out.DEoptim.fur <- optim(par = out.DEoptim$optim$bestmem, fn = obj, method = "Nelder-Mead")
> out.DEoptim.fur$value
[1] 0.1141564043
```

Note that the best value found by **DEoptim** was further improved by the use of a local search method. We ran **GenSA** as in the example below:

```
> set.seed(1234)
> fn.call <<- 0
> out.GenSA <- GenSA(fn = obj, lower = rep(0, 5), upper = rep(1, 5),
+                    control = list(smooth = FALSE, max.call = 3000))
> fn.call.GenSA <- fn.call
> out.GenSA$value
[1] 0.1141484884
> out.GenSA$counts
[1] 3000
> cat("GenSA call functions", fn.call.GenSA, "times.\n")
GenSA call functions 3000 times.
> wstar.GenSA <- out.GenSA$par
> wstar.GenSA <- wstar.GenSA / sum(wstar.GenSA)
> rbind(tickers, round(100 * wstar.GenSA, 2))
        [,1]    [,2]    [,3]   [,4]    [,5]
tickers "GE"    "IBM"   "JPM"  "MSFT"  "WMT"
        "18.92" "21.23" "8.33" "15.92" "35.6"
> 100 * (sum(wstar.GenSA * mu) - mean(mu))
[1] 0.03790568876
```

GenSA found a minimum 0.1141484884 within 3000 function calls. Though both **GenSA** and **DEoptim** find good approximations to the global minimum of the portfolio objective function, the result of **GenSA** is closer to the global minimum with fewer function calls.

Thomson problem in physics

The objective of the Thomson problem is to find the global energy minimum of N equal point charges on a sphere (Thomson, 1904). The Thomson model has been widely studied in physics (Erber and Hockney, 1991; Altschuler et al., 1994; Morris et al., 1996; Xiang et al., 1997; Xiang and Gong, 2000a; Levin and Arenzon, 2003; Wales and Ulker, 2006; Altschuler and Pérez-Garrido, 2006; Ji-Sheng, 2007; Wales et al., 2009). In the Thomson model, the energy of N point charges constrained on a sphere is

$$E = \frac{1}{2} \sum_{j \neq i} \frac{1}{\left| \vec{r}_i - \vec{r}_j \right|}. \tag{5}$$

For the Thomson problem, the number of metastable structures grows exponentially with N (Erber and Hockney, 1991). Moreover, the region containing the global minimum is often small compared with those of other minima (Morris et al., 1996). This adds to the difficulty of the Thomson problem and in part explains why it has served as a benchmark for global optimization algorithms in a number of previous studies (Erber and Hockney, 1991; Xiang et al., 1997; Xiang and Gong, 2000a; Altschuler and Pérez-Garrido, 2006). The Thomson problem has been tackled by several methods such as steepest descent (Erber and Hockney, 1991), constrained global optimization (CGO) (Altschuler et al., 1994), generalized simulated annealing algorithm (Xiang et al., 1997; Xiang and Gong, 2000a), genetic algorithms (Morris et al., 1996), and variants of Monte Carlo (Wales and Ulker, 2006; Wales et al., 2009). Typically, deterministic methods such as steepest descent with multiple starts can provide a good solution when there are fewer point charges (Erber and Hockney, 1991). When N is large, there are an enormous number of local minima since it increases exponentially with N (Erber and Hockney, 1991). For large N, the stochastic methods such as generalized simulated annealing, genetic algorithms, and variants of Monte Carlo, can give reasonable results (Xiang et al., 1997; Xiang and Gong, 2000a; Morris et al., 1996; Wales and Ulker, 2006; Wales et al., 2009).

We can define an R function for the Thomson problem.

```
> Thomson.fn <- function(x) {
+   fn.call <<- fn.call + 1
+   x <- matrix(x, ncol = 2)
+   y <- t(apply(x, 1, function(z) {
+    c(sin(z[1]) * cos(z[2]),
+      sin(z[1]) * sin(z[2]), cos(z[1]))\}))
+   n <- nrow(x)
+   tmp <- matrix(NA, nrow = n, ncol = n)
+   index <- cbind(as.vector(row(tmp)), as.vector(col(tmp)))
+   index <- index[index[, 1] < index[, 2], , drop=F]
+   rdist <- apply(index, 1, function(z) {
+     tmp <- 1/sqrt(sum((y[z[1], ] - y[z[2], ])^2))
+   })
+   res <- sum(rdist)
+   return(res)
+ }
```

In this example we chose a small number of point charges (12), since our purpose is only to show how to use **GenSA**. The global energy minimum of 12 equal point charges on the unit sphere is 49.16525306 with the corresponding configuration of an icosahedron (Erber and Hockney, 1991; Morris et al., 1996; Wales and Ulker, 2006).

We applied **DEoptim** with default settings to the Thomson problem.

```
> n.particles <- 12
> lower.T <- rep(0, 2 * n.particles)
> upper.T <- c(rep(pi, n.particles), rep(2 * pi, n.particles))
>
> options(digits = 10)
> set.seed(1234)
> sink("tmp.txt")
> fn.call <<- 0
> out.DEoptim <- DEoptim(fn = Thomson.fn, lower = lower.T, upper = upper.T)
> sink(NULL)
> fn.call.DEoptim <- fn.call
> out.DEoptim$optim[c(2, 3)]
$bestval
```

```
[1] 49.59590424

$nfeval
[1] 402

> cat("DEoptim call functions", fn.call.DEoptim, "times.\n")
DEoptim call functions 48240 times.
```

DEoptim used 48240 function calls to reach the function value 49.59590424, which is still far from the global minimum 49.16525306. Then we used a local search method to further refine the best value given by **DEoptim**.

```
> out.DEoptim_BFGS <- optim(par = out.DEoptim$optim$bestmem, fn = Thomson.fn,
+                           lower = lower.T, upper = upper.T, method = "L-BFGS-B")
> out.DEoptim_BFGS[c("value")]
$value
[1] 49.16525309
```

L-BFGS-B refined the function value to 49.16525309, which is close to the global minimum 49.16525306.

We applied **GenSA** to the Thomson problem with the same number of function calls as that of **DEoptim**:

```
> set.seed(1234)
> fn.call <<- 0
> out.GenSA <- GenSA(par = NULL, lower = lower.T, upper = upper.T,
+                    fn = Thomson.fn, control = list(max.call = fn.call.DEoptim))
> out.GenSA[c("value", "counts")]
$value
[1] 49.16525306

$counts
[1] 48240

> cat("GenSA call functions", fn.call, "times.\n")
GenSA call functions 48240 times.
```

We plotted the cumulative minimum of function value `Fmin` vs. number of function calls (`#fn.call`) for **GenSA** (green) and **DEoptim** (blue) in the Thomson problem in Figure 3. **GenSA** reached the global minimum (red asterisk) after 2791 function calls while **DEoptim** does not reach the global minimum after a much larger number of function calls (48240). Following the instruction for termination criteria in setion "The package GenSA", solving the Thomson problem with just 12 point charges is easy and we can stop GenSA much earlier by setting a smaller `max.call`.

For the Thomson problem, both **DEoptim** and **GenSA** converge to the global minimum. **GenSA** obtained more accurate results with fewer function calls than **DEoptim**.

GenSA relies on repeated evaluation of the objective function, so users who are interested in making **GenSA** run as fast as possible should ensure that evaluation of the objective function is also as efficient as possible.

Summary

Many R packages for solving optimization problems such as **DEoptim** and **rgenoud** have been developed and successfully used in the R community. Based on Tsallis statistics, an R package **GenSA** was developed for generalized simulated annealing. We propose **GenSA** be added to the tool kit for solving optimization problems in R. The package **GenSA** has proven to be a useful tool for solving global optimization problems. The applicability of **GenSA** was demonstrated with a standard test function, the Rastrigin function, a simple example of a stylized non-convex portfolio risk contribution allocation, and the Thomson problem in physics. As no single method is the best for all the optimization problems, **GenSA** can serve as a complementary tool to, rather than a replacement for, other widely used R packages for optimization. We hope that the availability of **GenSA** will allow users to have more choices when they process difficult objective functions. The results are based on **GenSA** version 1.1.3, **DEoptim** version 2.2.1, and R 2.15.2.

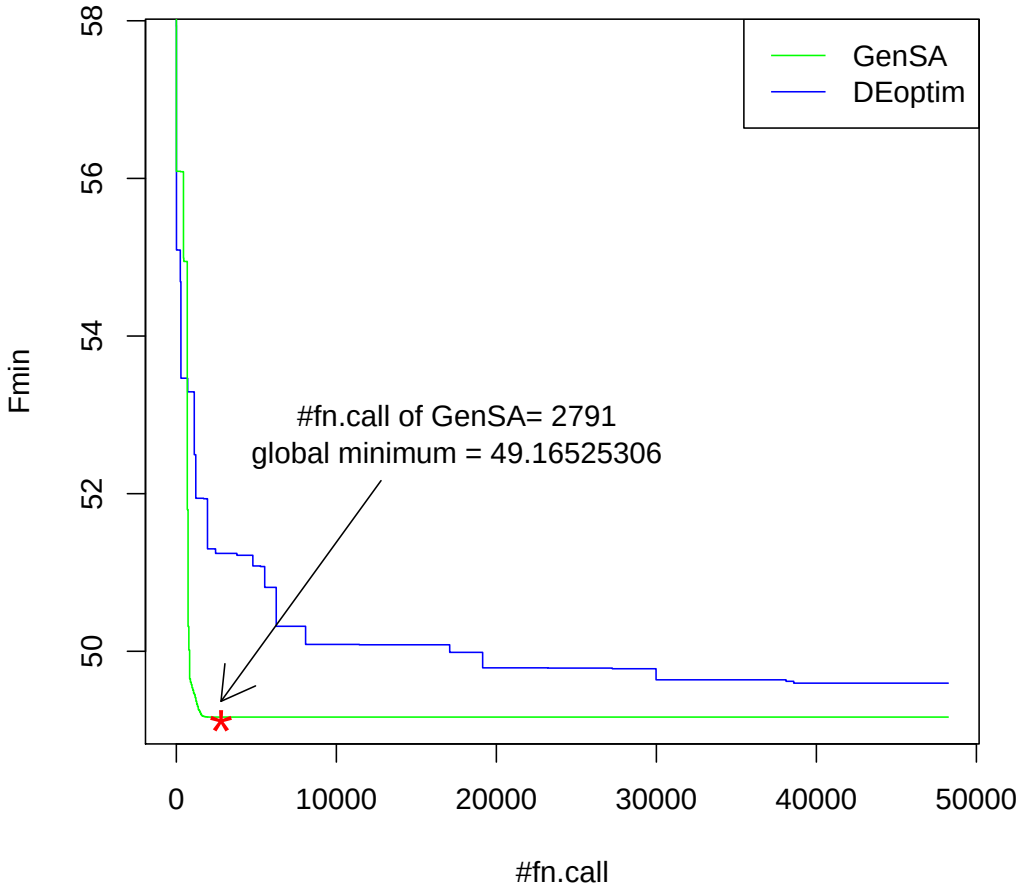


Figure 3: Cumulative minimum of function value `Fmin` vs. number of function calls (`#fn.call`) for **GenSA** (green) and **DEoptim** (blue) in the Thomson problem (number of point charges is 12). **GenSA** reaches the global minimum (red asterisk) after 2791 function calls while **DEoptim** does not reach the global minimum after a much larger number of function calls (48240).

Acknowledgements

This work would not be possible without the R open source software project. We would like to thank our colleague Florian Martin for inspirational discussions. We acknowledge our colleague Lynda Conroy-Schneider for editing our manuscript. We would also like to thank the reviewers for giving such constructive comments which substantially improved the quality of the manuscript.

Bibliography

F. P. Agostini, D. D. O. Soares-Pinto, M. A. Moret, C. Osthoff, and P. G. Pascutti. Generalized simulated annealing applied to protein folding studies. *Journal of Computational Chemistry*, 27:1142–1155, 2006. [p13]

E. Altschuler and A. Pérez-Garrido. Defect-free global minima in thomson's problem of charges on a sphere. *Physical Review E*, 73(3):036108, 2006. [p23]

E. L. Altschuler, T. J. Williams, E. R. Ratner, F. Dowla, and F. Wooten. Method of constrained global optimization. *Phys. Rev. Lett.*, 72:2671–2674, Apr 1994. doi: 10.1103/PhysRevLett.72.2671. URL http://link.aps.org/doi/10.1103/PhysRevLett.72.2671. [p23]

D. Ardia, K. Boudt, P. Carl, K. M. Mullen, and B. G. Peterson. Differential evolution with DEoptim. *The R Journal*, 3, 2011. [p14, 16, 21, 22]

R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. [p17]

J. O. Cerdeira, P. D. Silva, J. Cadima, and M. Minhoto. *subselect: Selecting variable subsets*, 2011. URL http://CRAN.R-project.org/package=subselect. R package version 0.11-1. [p14]

D. Cvijovic and J. Klinowski. Taboo search: An approach to the multiple minima problem. *Science*, 267:664–666, 1995. [p13]

D. Cvijovic and J. Klinowski. Taboo search: An approach to the multiple-minima problem for continuous functions. In R. H. Pardalos Pm, editor, *Handbook of Global Optimization*, volume 2, pages 387–406. Kluwer Academic Publisher, 2002. [p13]

T. Erber and G. M. Hockney. Equilibrium configurations of n equal charges on a sphere. *Journal of Physics A: Mathematical and General*, 24:L1369, 1991. [p23]

E. Figielska. A genetic algorithm and a simulated annealing algorithm combined with column generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources. *Computers & Industrial Engineering*, 56(1):142–151, 2009. [p14]

D. Fouskakis and D. Draper. Stochastic optimization: a review. *International Statistical Review*, 70(3): 315–349, 2002. [p13]

F. Glover, E. Taillard, and E. Taillard. A user's guide to tabu search. *Annals of operations research*, 41(1): 1–28, 1993. [p13]

V. D. Guire, M. Caron, N. Scott, C. Menard, M. F. Gaumont-Leclerc, P. Chartrand, F. Major, and G. Ferbeyre. Designing small multiple-target artificial rnas. *Nucleic Acids Research*, 38(13):e140, 2010. [p13]

J. H. Holland. Adaptation in natural and artificial systems. *The University of Michigan Press, Ann Arbor*, 1975. [p13]

C. Ji-Sheng. Ground state energy of unitary fermion gas with the thomson problem approach. *Chinese Physics Letters*, 24:1825, 2007. [p23]

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220: 671–680, 1983. [p13, 14]

Y. Levin and J. Arenzon. Why charges go to the surface: A generalized thomson problem. *EPL (Europhysics Letters)*, 63:415, 2003. [p23]

W. Mebane Jr and J. Sekhon. Genetic optimization using derivatives: the rgenoud package for r. *Journal of Statistical Software*, 42(11):1–26, 2011. [p14, 16, 17, 18]

J. R. Morris, D. M. Deaven, and K. M. Ho. Genetic-algorithm energy minimization for point charges on a sphere. *Physical Review B*, 53:1740–1743, 1996. [p23]

K. Mullen, D. Ardia, D. Gil, D. Windover, and J. Cline. Deoptim: An r package for global optimization by differential evolution. *Journal of Statistical Software*, 40(6):1–26, 2011. URL http://www.jstatsoft.org/v40/i06/. [p14, 16, 21]

L. Murphy. *likelihood: Methods for maximum likelihood estimation*, 2008. R package version 1.4. [p14]

J. C. Nash. *Compact numerical methods for computers: linear algebra and function minimisation*. Taylor & Francis, 1990. ISBN 978-0852743195. [p17]

S. Pavlidis, A. Payne, and S. Swift. Multi-membership gene regulation in pathway based microarray analysis. *Algorithms for Molecular Biology*, 6(1):22, 2011. ISSN 1748-7188. doi: 10.1186/1748-7188-6-22. URL http://www.almob.org/content/6/1/22. [p14]

M. J. D. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Report No. DAMTP, 2009/NA06, Centre for Mathematical Sciences, University of Cambridge*, 2009. [p14, 17]

P. Serra, A. F. Stanton, and S. Kaisb. Comparison study of pivot methods for global optimization. *Journal of Chemical Physics*, 106:7170–7177, 1997. [p13, 14, 18]

P. Sólymos. dclone: Data cloning in R. *The R Journal*, 2(2):29–37, 2010. URL http://journal.r-project.org/. [p14]

R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997. [p13]

H. Szu and R. Hartley. Fast simulated annealing. *Physics Letters A*, 122:157–162, 1987. [p14]

S. Theussl. *CRAN Task View: Optimization and Mathematical Programming*, 04 2011. URL http://cran.r-project.org/web/views/Optimization.html. [p14]

D. Thompson and G. Bilbro. Comparison of a genetic algorithm with a simulated annealing algorithm for the design of an atm network. *Communications Letters, IEEE*, 4(8):267–269, 2000. [p14]

J. J. Thomson. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *Philosophical Magazine*, 7:237–265, 1904. [p23]

C. Tsallis and D. A. Stariolo. Generalized simulated annealing. *Physica A*, 233:395–406, 1996. [p14, 15]

J. VanDerWal and S. Januchowski. *ConsPlan: Conservation Planning Tools*, 2010. R package version 0.1. [p14]

D. Wales and S. Ulker. Structure and dynamics of spherical crystals characterized for the thomson problem. *Physical Review B*, 74(21):212101, 2006. [p23]

D. Wales, H. McKay, and E. Altschuler. Defect motifs for spherical topologies. *Physical Review B*, 79 (22):224115, 2009. [p23]

Y. Xiang and X. G. Gong. Efficiency of generalized simulated annealing. *Physical Review E*, 62:4473–4476, 2000a. [p13, 14, 15, 23]

Y. Xiang and X. G. Gong. Generalized simulated annealing method and its application. *Progress In Physics*, 20:319–334, 2000b. [p14]

Y. Xiang, D. Y. Sun, W. Fan, and X. G. Gong. Generalized simulated annealing algorithm and its application to the thomson model. *Physics Letters A*, 233:216–220, 1997. [p13, 14, 15, 23]

Y. Xiang, D. Y. Sun, and X. G. Gong. Generalized simulated annealing studies on structures and properties of ni_n (n=2-55) clusters. *Journal of Physical Chemistry A*, 104:2746–2751, 2000. [p13, 15]

X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, 1999. [p17]

C. Zhu, R. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997. [p17]

Yang Xiang
Philip Morris International R&D
Philip Morris Products S.A.
Quai Jeanrenaud 5
2000 Neuchatel
Switzerland
Yang.Xiang@pmi.com

Sylvain Gubian
Philip Morris International R&D
Philip Morris Products S.A.
Quai Jeanrenaud 5
2000 Neuchatel
Switzerland
Sylvain.Gubian@pmi.com

Brian Suomela
Philip Morris International R&D
Philip Morris Products S.A.
Quai Jeanrenaud 5
2000 Neuchatel
Switzerland
Brian.Suomela@pmi.com

Julia Hoeng
Philip Morris International R&D
Philip Morris Products S.A.
Quai Jeanrenaud 5
2000 Neuchatel
Switzerland
Julia.Hoeng@pmi.com

Multiple Factor Analysis for Contingency Tables in the FactoMineR Package

by Belchin Kostov, Mónica Bécue-Bertaut and François Husson

Abstract We present multiple factor analysis for contingency tables (MFACT) and its implementation in the **FactoMineR** package. This method, through an option of the `MFA` function, allows us to deal with multiple contingency or frequency tables, in addition to the categorical and quantitative multiple tables already considered in previous versions of the package. Thanks to this revised function, either a multiple contingency table or a mixed multiple table integrating quantitative, categorical and frequency data can be tackled.

The **FactoMineR** package (Lê et al., 2008; Husson et al., 2011) offers the most commonly used principal component methods: principal component analysis (PCA), correspondence analysis (CA; Benzécri, 1973), multiple correspondence analysis (MCA; Lebart et al., 2006) and multiple factor analysis (MFA; Escofier and Pagès, 2008). Detailed presentations of these methods enriched by numerous examples can be consulted at the website `http://factominer.free.fr/`.

An extension of the `MFA` function that considers contingency or frequency tables as proposed by Bécue-Bertaut and Pagès (2004, 2008) is detailed in this article.

First, an example is presented in order to motivate the approach. Next, the mortality data used to illustrate the method are introduced. Then we briefly describe multiple factor analysis (MFA) and present the principles of its extension to contingency tables. A real example on mortality data illustrates the handling of the `MFA` function to analyse these multiple tables and, finally, conclusions are presented.

Motivation

MFACT was initially conceived to deal with international surveys including open-ended questions answered in different languages such as the survey designed by Akuto (1992) for comparing the dietary habits in Tokyo, Paris and New York. Three samples of individuals, each of about thousand respondents, were taken in the three cities and answered a classical questionnaire and also the open-ended question: "Which dishes do you like and eat often?". So three groups of free-text answers in three different languages have to be analyzed.

If a unique sample (for example, New York) were analysed, the table crossing gender $\times$ age and words could be considered to see how the dietary habits change with age and gender. The age variable is divided into clusters such as under 30, $30-50$, and over 50.

For this kind of table, correspondence analysis is a reference method (Lebart et al., 1998). Through a superimposed map of the categories and word-dishes, CA allows us to visualize

1. The similarities between the gender $\times$ age categories: two categories are closer if they often eat the same dishes
2. The similarities between dishes: two dishes are closer if they are often chosen by the same categories;
3. The attractions (respectively, the repulsions) between categories and dishes: a category is at the pseudo-centroid of the word-dishes used in the answers belonging to it; a word-dish is at the pseudo-centroid of the categories that mention it.

In order to analyse and to compare several samples (New York, Tokyo and Paris), we can consider the frequency table that juxtaposes row-wise the three contingency tables. This large frequency table can then be analysed through MFACT (Pagès and Bécue-Bertaut, 2006). In this case, we can answer the following questions:

1. Which are the gender $\times$ age categories that are globally similar (through all the samples)?
2. Which are the similarities between dishes?
3. Which dishes are associated with which gender $\times$ age categories?

These three questions are the same as when only one sample was considered, but considering now all the samples. In order to simultaneously consider the three samples, MFACT centres each table on its own centroid and balances the influence of each sample in the global analysis to prevent one table playing a dominating role. Moreover, MFACT allows us to compare the three samples in a common framework:

- The gender × age category structures can be represented as issued from every sample thanks to the partial representations: it indicates how similar or different are the category structures from one city to another, from the point of view of the dish preferences.
- The methodology also allows us to study the similarities between cities: two cities are closer if the attractions (respectively, the repulsions) between categories and dishes induced by each table are similar.

The type of result provided by MFACT makes this method useful to globally analyze and compare several contingency tables. It is able to deal with any number of tables and could be used to compare the dietary habits in a large number of cities.

The dataset

In this paper, we illustrate the method through data on "the causes of mortality" provided by the "Centre d'épidémiologie sur les causes médicales de décès – Cépidc" (http://www.cepidc.vesinet.inserm.fr). These data concern the mortality data in France from 1979 to 2006. The registration of the causes of mortality is mainly motivated by prevention: identifying and quantifying the causes of mortality in order to take action to reduce avoidable mortality. Therefore, these data are used to produce one of the health indicators most commonly used.

Our aim is to study the evolution of the causes of mortality from 1979 to 2006. For each year, a crossed table contains 62 mortality causes (in rows) and age intervals (in columns). At the intersection of a row-mortality cause and a column-age interval, the counts of deaths corresponding to this cause and this age interval during this year is given. Note that the cause AIDS has been removed, since this cause did not exist in the mortality registers in 1979. Likewise, sudden infant death syndrome (SIDS) and perinatal infection, specific causes of mortality for children, have been removed.

The data are read from the **FactoMineR** package:

```
> library(FactoMineR)
> data(mortality)
```

Multiple factor analysis for contingency tables (MFACT)

Recall on multiple factor analysis

Multiple factor analysis (Escofier and Pagès, 2008) deals with a multiple table, composed of groups of either quantitative or categorical variables. MFA balances the influence of the groups on the first principal dimension by dividing the weights of the variables/columns of a group by the first eigenvalue of the separate analysis of this group (PCA or MCA depending on the type of the variables). The highest axial inertia of each group is standardized to 1.

MFA provides the classical results of principal component methods. PCA characteristics and interpretation rules are kept for the quantitative groups and those of MCA for the categorical groups. MFA offers tools for comparing the different groups such as the partial representation of the rows. This representation allows us to compare the typologies provided by each group in a common space. A graphic of the groups allows us to globally compare the groups and to evaluate if the relative positions of the rows are globally similar from one group to another. It also permits the comparison of the partial groups and assess whether they provide the same information. Another graph gives the correlations between the dimensions of the global analysis (the MFA dimensions) and each separate analysis (the PCA dimensions for quantitative groups, the MCA dimensions for categorical groups and the CA dimensions for frequency groups).

Multiple tables and notation

The multiple contingency table X_G (the causes of mortality in 1979 and 2006), of dimensions $I \times J$ (62 mortality causes × 18 age intervals in total), juxtaposes row-wise several contingency tables $X_1, ..., X_t, ..., X_T$ (one table for each year, in the example $T = 2$) of dimension $I \times J_t$ (62 mortality causes × 9 age intervals) for table t (see Figure 1). Columns can differ from one table to another (which is not the case in this example).

X_G is transformed into a proportion table (proportion of the causes of mortality for each age interval), globally computed on all the tables. Thus p_{ijt} is the proportion of row-mortality cause i ($i = 1, ..., I$) for column-age interval j ($j = 1, ..., J_t$) of table t ($t = 1, ..., T$); $\sum_{ijt} p_{ijt} = 1$. The row and column margins of table X_G are respectively $p_{i\bullet\bullet} = \sum_{jt} p_{ijt}$ and $p_{\bullet jt} = \sum_i p_{ijt}$. The intra-table

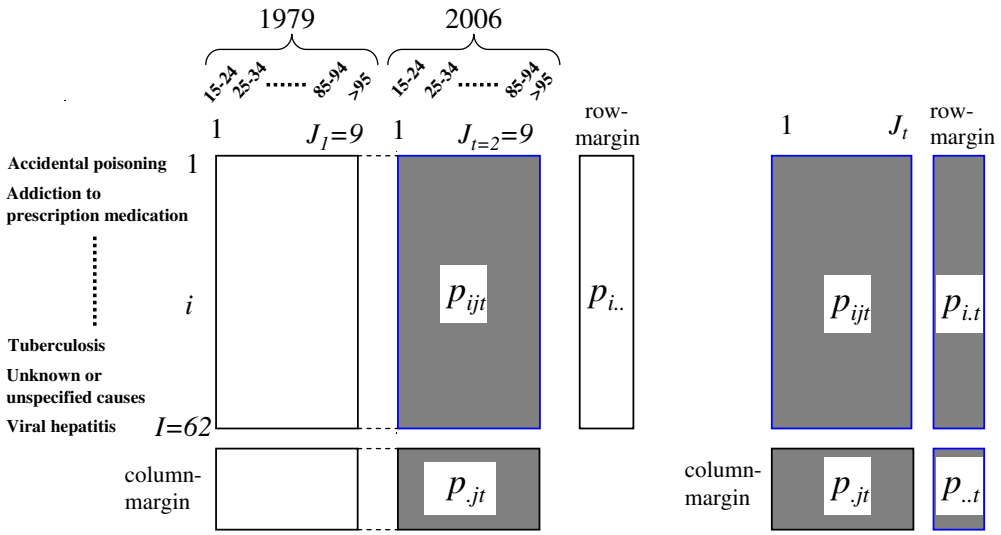


Figure 1: Multiple contingency table. T contingency tables with the same rows are juxtaposed row-wise. On the left, the global table and margins; on the right the table t and its margins. In the example, $T = 2$, $I = 62$, $J_1 = 9$, $J_2 = 9$.

independence model is considered (relations between causes of mortality and age intervals in each year). It is filled in a matrix with general term

$$\left(\frac{p_{i\bullet t}}{p_{\bullet\bullet t}}\right) \times \left(\frac{p_{\bullet jt}}{p_{\bullet\bullet t}}\right)$$

where $p_{i\bullet t} = \sum_j p_{ijt}$ is the row margin of table t (the percentage of the causes of mortality i in year t with respect to the sum of causes of mortality for all years) and $p_{\bullet\bullet t} = \sum_{ij} p_{ijt}$ is the sum of the terms of table t inside table X_G (the percentage of the sum of causes of mortality in year t with respect to the sum of causes of mortality for all years). Then table Z whose general term is the weighted residual with respect to the intra-table independence model is built. Note that Z is divided into T subtables Z_t for $t = 1, ..., T$.

$$Z = \frac{p_{ijt} - \left(\frac{p_{i\bullet t}}{p_{\bullet\bullet t}}\right) p_{\bullet jt}}{p_{i\bullet\bullet} \times p_{\bullet jt}} = \frac{1}{p_{i\bullet\bullet}} \left(\frac{p_{ijt}}{p_{\bullet jt}} - \frac{p_{i\bullet t}}{p_{\bullet\bullet t}}\right)$$

The algorithm

MFACT consists of a classical MFA applied to the multiple table Z assigning the weight $p_{i\bullet\bullet}$ to the row i ($i = 1, ..., I$) and the weight $p_{\bullet jt}$ to the column jt ($j = 1, ..., J_t$, $t = 1, ..., T$). Thus the observed proportions are compared to those corresponding to the intra-table independence model. This model neutralizes the differences between the separate average column profiles.

It is easy to verify that the weighted rows are centred as well as the weighted columns of each table t. The influence of each subtable in the global analysis is balanced in a MFA-like way. MFA classical results are obtained and CA characteristics and interpretation rules are kept.

The **FactoMineR** package (from version 1.16) includes MFACT in the `MFA` function.

Results

The analysis of one contingency table by means of CA was discussed in Husson et al. (2011). If we want to deal with two contingency tables, one of the tables could be favoured and chosen as active while the other would be considered as supplementary. However, that would lead to principal dimensions only based on the active contingency table. When both tables, that is, the mortality data of both years, have to play a balanced role to enhance the evolution of the mortality causes, MFACT turns out to be necessary.

We have used MFACT to evaluate the evolution of the mortality causes considering changes at

the age profiles (between 1979 and 2006). MFACT preserves the internal structure of the relationships between causes of mortality and age intervals in each table and balances their influences on the global representation (Bécue-Bertaut and Pagès, 2004).

Implementation in FactoMineR

MFA is performed on the multiple table "causes of mortality" as follows:

```
> mfa <- MFA(mortality, group=c(9, 9), type=c("f", "f"), name.group=c("1979", "2006"))
```

In this call, only few arguments were used. The `group` argument specifies that each table has 9 columns, `type` specifies the type of the tables, here frequency tables, `name.group` gives the name of each table. By default, all the individuals are considered as active (the `ind.sup` argument is equal to `NULL`), all the groups are active (`num.group.sup` argument is equal to `NULL`) and the plots will be drawn for the first two dimensions.

Outputs

The outputs of MFA are both a series of numerical indicators (results of the separate analysis, eigenvalues, coordinates, contributions, correlations) and a series of graphics (individuals/rows, superimposed representation, variables/columns, partial axes, groups). Most of them are presented hereinafter along with an interpretation of the results.

Eigenvalues

The sequence of eigenvalues identifies two dominating dimensions that, together, account for 81.69% of the total inertia (see below). That leads to focus on the first principal plane.

```
> round(mfa$eig,3) [1:4,]
        eigenvalue   percentage cumulative \%
                    of variance  of variance
comp 1       1.790       52.420       52.420
comp 2       0.999       29.269       81.689
comp 3       0.262        7.659       89.348
comp 4       0.149        4.367       93.715
```

We recall that in MFA the first eigenvalue varies between 1 and the number of groups (two in this case). A first eigenvalue close to the maximum means that the first dimension of the separate analyses (here the CAs on each contingency table) are very similar. Thus, the value of the first eigenvalue (equal to 1.79) indicates that the first principal component is an important dispersion dimension in both tables, and is similar to the first axes of the two separate CAs. Then, we can say that the similarity between both groups justifies their simultaneous analysis and that there are enough differences to resort to a method able to highlight common and specific features.

Representation of rows and columns

Figure 2 visualizes the age intervals columns on the first principal plane. The trajectories of the age intervals of both years are drawn in different colours to ease the comparison.

```
> plot(mfa, choix="freq", invisible="ind", habillage="group")
> lines(mfa$freq$coord[1:9, 1], mfa$freq$coord[1:9, 2], col="red")
> lines(mfa$freq$coord[10:18, 1], mfa$freq$coord[10:18, 2], col="green")
```

The first dimension perfectly ranks the age intervals, opposing the youngest to the oldest. The second dimension opposes the extreme to the medium-age intervals. The homologous age intervals corresponding to 1979 and 2006 are very close. Both age trajectories are almost identical. This means that, the relationship between mortality causes and age intervals are very similar in 1979 and 2006. However, we will see hereinafter some interesting differences provided by the comparison of the groups.

In both years, the youngest intervals differ more from one another that the oldest: the distances between two consecutive age intervals are longer in the case of the youngest ages (Figure 2). That indicates that the predominant causes of mortality in young people change rapidly with age.

The visualization of the mortality causes on the principal plane shows that some causes are clearly separated from the others (Figure 3).

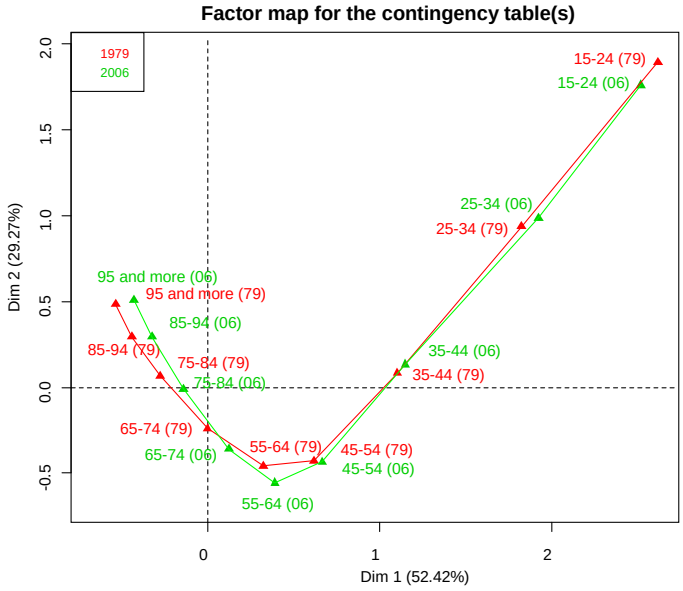


Figure 2: Trajectories of the age intervals on the first principal plane.

```
> sel <- c(2, 8:10, 15, 38, 58)
> plot(mfa, lab.ind=FALSE, habillage="group")
> text(mfa$ind$coord[sel, 1], mfa$ind$coord[sel, 2], rownames(mortality)[sel],
       pos=c(2, 2, 2, 2, 4, 2, 4))
```

This finding makes sense when related to the age intervals through the usual transition relationships. Indeed, an age interval is attracted (or repelled) by the causes that are more (or less) associated with it than if there were independence between causes and age intervals of either 1979 or 2006.

An *arch effect*, also called *Guttman effect*, is identified on the first principal plane both on rows and columns (Figures 2 and 3), pointing out that the phenomenon under study is mainly unidimensional and that both rows and columns share ranking. From the transition relationships, we can conclude that the first component ranks the causes from the "young causes" (on the right) to the "old causes" (on the left). The second component opposes the causes that affect the extreme age intervals to those concerning, mainly, the medium age intervals.

The causes highly associated with young age are very specific (Figure 3): complications in pregnancy and childbirth, addiction to prescription medication, road accidents, meningococcal disease, homicides, congenital defects of the nervous system and congenital defects of the circulatory system. They are responsible of a high proportion of the young deaths: 30% of the deaths under 35 are due to these causes. At the same time, nearly half of the deaths due to these causes are among the young: from a total count of 17211 due to these causes, 7891 are under 35.

These causes, except road accidents, are relatively unimportant within all mortality counts and thus their contributions to the construction of the dimensions are low:

```
> round(mfa$ind$contr[c(2, 8:10, 15, 38, 58), 1:2], 3)
Addiction to prescription medication  0.998  0.448
Complications in pregnancy & childb.  0.685  0.527
Congenital defects circulatory system 0.692  0.176
Congenital defects nervous system     0.179  0.070
Homicides                             1.802  0.657
Meningococcal disease                 0.105  0.084
Road accidents                       34.295 23.364
```

In the case of the road accidents, a general trend is observed: as age increases, death counts caused by road accidents decrease. This phenomenon is observed in both 1979 and 2006.

```
> mortality[58, 1:9]  # road accidents in 1979
 15-24   25-34   35-44   45-54   55-64    65-74   75-84   85-94   95 and more
  3439    1666    1195    1328    966      1117     757     135             4
> mortality[58, 10:18] # road accidents in 2006
```

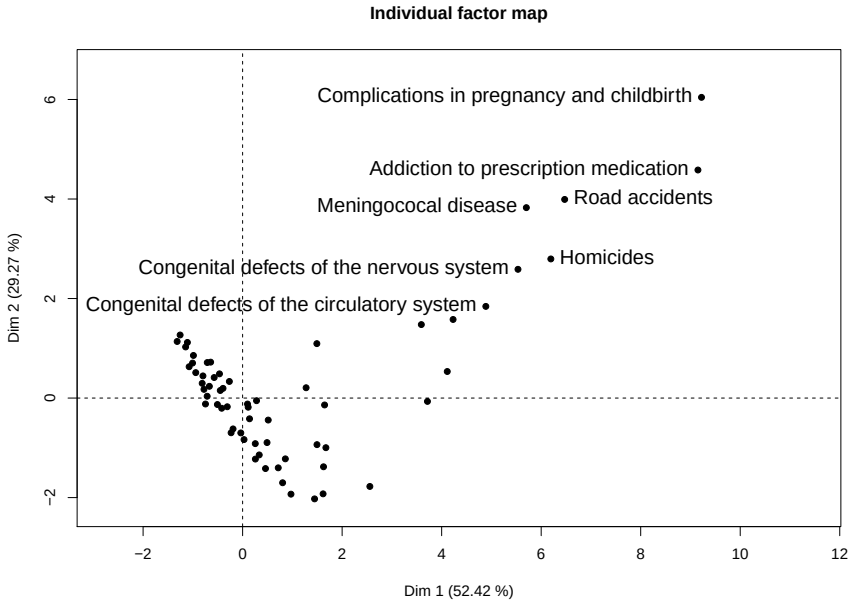


Figure 3: Mortality causes representation on the first principal plane. Only the "young" causes are labelled.

15-24	25-34	35-44	45-54	55-64	65-74	75-84	85-94	95 and more
1214	785	646	599	443	362	454	137	8

Dimensions of the separate analyses

The dimensions of the separate analyses can be represented through their correlations with the MFA dimensions. The first and second dimensions of the global analysis are very correlated with the first and second dimensions of the separate analyses (Figure 4).

Synthetic representation of the groups

As there are only two groups, their representation provides little information. However, in the case of a high number of groups, this kind of representation would be very useful for a global comparison.

Figure 5 shows that both groups have coordinates on dimension 1 that are very close to 1 showing that they are sensitive to the age ranking as reflected by this dimension. There are some differences between the two groups on dimension 2: the oppositions between causes, highlighted on the second dimension, are slightly more pronounced in 2006.

Superimposed representation

The superimposed representation of the partial rows (Figure 6) shows the more important changes between 1979 and 2006.

```
> sel <- c(2, 10, 41, 58)
> plot(mfa, lab.ind=FALSE, habillage="group", partial=rownames(mortality)[sel])
> text(mfa$ind$coord[sel,1], mfa$ind$coord[sel,2], rownames(mortality)[sel],pos=4)
```

Some important changes concerning the distribution of mortality causes among the age intervals between 1979 and 2006 are identified. Addiction to prescription medication is the cause of death showing the highest difference. It moves from close to the centroid (1979) to a position with a high coordinate (2006) on the first dimension: deaths related to this cause concern younger people more in 2006 than in 1979.

Table 1 ratifies this change. In 1979, about 60% of all deaths due to addiction to prescription medication are over 44. However, in 2006, 80% of deaths associated to this cause are between 25 and 44.

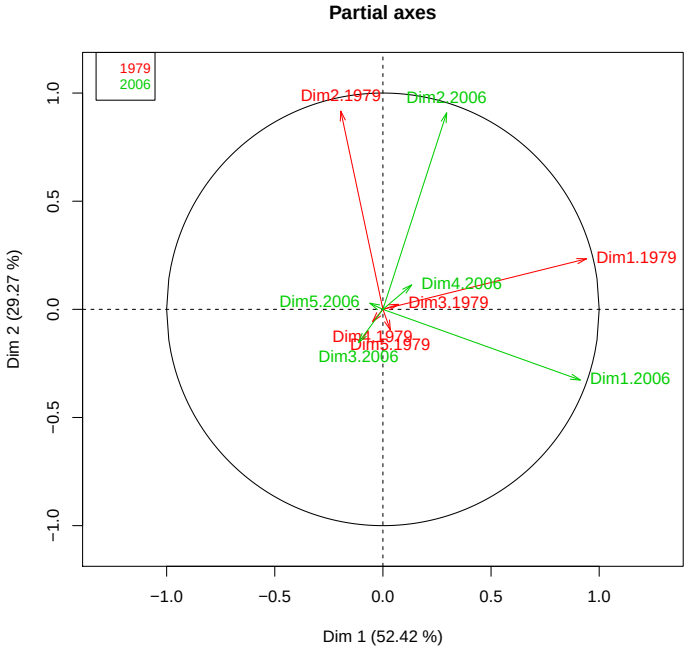


Figure 4: Dimensions of the separate analyses on the first principal plane issued from MFA.

Addiction to prescription medication	[15-24]	[25-34]	[35-44]	[45-54]	[55-64]	[65-74]	[75-84]	[85-94]	[95 or more]	Total
1979	7	4	2	6	6	2	6	0	0	33
%	21.2	12.1	6.1	18.2	18.2	6.1	18.2	0	0	100
2006	18	77	72	15	4	2	1	0	0	189
%	9.5	40.7	38.1	7.9	2.1	1.1	0.5	0	0	100

Table 1: Counts and percentages of deaths related to addiction to prescription medication in 1979 and 2006.

Conclusions

The study of mortality causes in France in 1979 and 2006 has shown the kind of results offered by MFA extended to deal with a multiple contingency table.

Concerning these data, we can conclude that the mortality structure by age changes very little from 1979 to 2006, apart from some interesting nuances. In both years, younger age is associated with very specific mortality causes. The most important change is due to addiction to prescription medication. This cause turns to be, at the same time, more important (with respect to the total count) and associated more to young ages in 2006 compared to 1979.

MFACT can be used in many different fields, such as text mining, public health, sensometrics, etc. In the example used to illustrate the motivation, MFACT has been applied to compare the favourite menus in different countries (Pagès and Bécue-Bertaut, 2006). To give a few examples, MFACT has also been used to cluster the Spanish regions depending on their mortality structure (Bécue-Bertaut et al., 2011), and to characterize food products from free-text descriptions (Kostov et al., 2011).

The `MFA` function offers options, tools and graphical outputs that ease the interpretation of the results. Furthermore, this function allows the analysis of multiple tables with a mixture of quantitative, categorical and frequency groups as detailed in Bécue-Bertaut and Pagès (2008). This latter possibility is largely used in studies such as:

- The relationship between the bacterial community structures and hydrogeochemical conditions in a groundwater study (Imfeld et al., 2011).
- The influence of package shape and colour on consumer expectations of milk desserts using word association and conjoint analysis (Ares and Deliza, 2010).
- Customer portfolio composition (Abascal et al., 2010).
- The answers to an open-ended question to identify drivers of liking of milk desserts (Ares et al.,

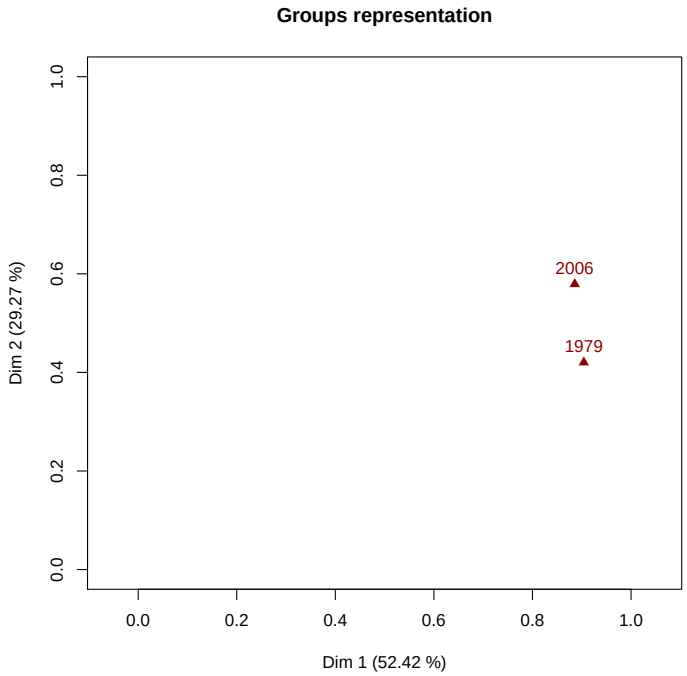


Figure 5: Synthetic representation of the groups on the first and second dimensions.

2010).

It is possible to enrich the analysis using supplementary variables, including quantitative, qualitative and frequency variables.

Summary

MFACT is a useful method to analyze multiple contingency tables. Here, we present an application of this method to French mortality data to compare the relationships between mortality causes and age intervals in two different years. We show how this method helps to compare the same information on two different occasions. It is possible to also apply MFACT to multiple tables integrating frequency, categorical and quantitative groups of variables observed on the same individuals.

Bibliography

E. Abascal, I. G. Lautre, and F. Mallor. Tracking customer portfolio composition: A factor analysis approach. *Applied Stochastic Models in Business and Industry*, 26:535–550, 2010. [p35]

H. Akuto. *International Comparison of Dietary Cultures*. Nihon Keizai Shimbun, 1992. [p29]

G. Ares and R. Deliza. Studying the influence of package shape and colour on consumer expectations of milk desserts using word association and conjoint analysis. *Food Quality and Preference*, 21:930–937, 2010. [p35]

G. Ares, R. Deliza, C. Barreiro, A. Giménez, and A. Gámbaro. Use of an open-ended question to identify drivers of liking of milk desserts. Comparison with preference mapping techniques. *Food Quality and Preference*, 21:286–294, 2010. [p35]

M. Bécue-Bertaut and J. Pagès. A principal axes method for comparing multiple contingency tables: MFACT. *Computational Statistics and Data Analysis*, 45:481–503, 2004. [p29, 32]

M. Bécue-Bertaut and J. Pagès. Multiple factor analysis and clustering of a mixture of quantitative, categorical and frequency data. *Computational Statistics and Data Analysis*, 52:3255–3268, 2008. [p29, 35]

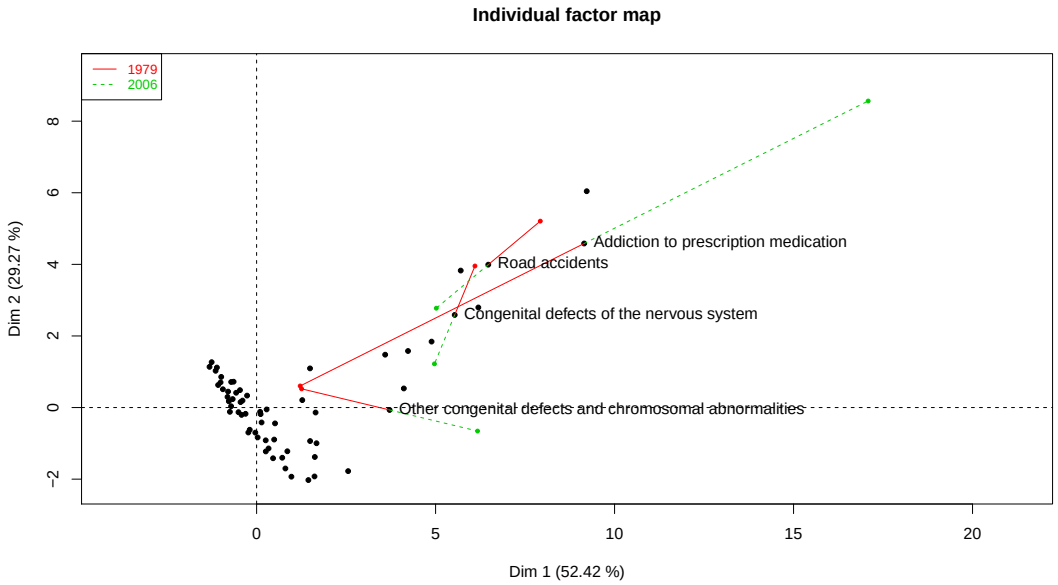


Figure 6: Excerpt of the superimposed representation of the partial and global mortality causes.

M. Bécue-Bertaut, M. Guillén, and J. Pagès. Clasificación de las regiones españolas según sus patrones de mortalidad. In *Métodos Cuantitativos en Economía del Seguro del Automóvil*, pages 97–111. Universidad de Barcelona, 2011. [p35]

J. Benzécri. *Analyse des Données*. Dunod, 1973. [p29]

B. Escofier and J. Pagès. *Analyses Factorielles Simples et Multiples*. Dunod, 4th edition, 2008. [p29, 30]

F. Husson, S. Lê, and J. Pagès. *Exploratory Multivariate Analysis by Example Using R*. Chapman & Hall / CRC Press, 2011. [p29, 31]

G. Imfeld, H. Pieper, N. Shani, P. Rossi, M. Nikolausz, I. Nijenhuis, H. Paschke, H. Weiss, and H. H. Richnow. Characterization of groundwater microbial communities, dechlorinating bacteria, and in situ biodegradation of chloroethenes along a vertical gradient. *Water, Air, & Soil Pollution*, 221(1–4): 107–122, 2011. [p35]

B. Kostov, M. Bécue-Bertaut, J. Pagès, M. Cadoret, J. Torrens, and P. Urpi. Verbalisation tasks in Hall test sessions. Presented at the "International Classification Conference (ICC)", St. Andrews, UK, 2011. [p35]

S. Lê, J. Josse, and F. Husson. FactoMineR: An R package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18, 3 2008. ISSN 1548-7660. URL http://www.jstatsoft.org/v25/i01. [p29]

L. Lebart, A. Salem, and L. Berry. *Exploring Textual Data*. Kluwer Academic Publishers, 1998. [p29]

L. Lebart, M. Piron, and A. Morineau. *Statistique Exploratoire Multidimensionnelle*. Dunod, 2006. [p29]

J. Pagès and M. Bécue-Bertaut. Multiple factor analysis for contingency tables. In M. Greenacre and J. Blasius, editors, *Multiple Correspondence Analysis and Related Methods*, pages 299–326. Chapman & Hall / CRC Press, 2006. [p29, 35]

Belchin Kostov
Transverse group for research in primary care, IDIBAPS Primary Health Care Center Les Corts, CAPSE
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

Hypothesis Tests for Multivariate Linear Models Using the car Package

by John Fox, Michael Friendly, and Sanford Weisberg

Abstract The *multivariate linear model* is

$$\underset{(n\times m)}{\mathbf{Y}} = \underset{(n\times p)}{\mathbf{X}}\underset{(p\times m)}{\mathbf{B}} + \underset{(n\times m)}{\mathbf{E}}$$

The multivariate linear model can be fit with the `lm` function in R, where the left-hand side of the model comprises a matrix of response variables, and the right-hand side is specified exactly as for a univariate linear model (i.e., with a single response variable). This paper explains how to use the `Anova` and `linearHypothesis` functions in the **car** package to perform convenient hypothesis tests for parameters in multivariate linear models, including models for repeated-measures data.

Basic ideas

The *multivariate linear model* accommodates two or more *response* variables. The theory of multivariate linear models is developed very briefly in this section, which is based on Fox (2008, Sec. 9.5). There are many texts that treat multivariate linear models and multivariate analysis of variance (*MANOVA*) more extensively: The theory is presented in Rao (1973); more generally accessible treatments include Hand and Taylor (1987) and Morrison (2005). A good brief introduction to the MANOVA approach to repeated-measures may be found in O'Brien and Kaiser (1985), from which we draw an example below. Winer (1971, Chap. 7) presents the traditional univariate approach to repeated-measures ANOVA.

The multivariate general linear model is

$$\underset{(n\times m)}{\mathbf{Y}} = \underset{(n\times p)}{\mathbf{X}}\underset{(p\times m)}{\mathbf{B}} + \underset{(n\times m)}{\mathbf{E}}$$

where $\mathbf{Y}$ is a matrix of n observations on m response variables; $\mathbf{X}$ is a model matrix with columns for p regressors, typically including an initial column of 1s for the regression constant; $\mathbf{B}$ is a matrix of regression coefficients, one column for each response variable; and $\mathbf{E}$ is a matrix of errors. The contents of the model matrix are exactly as in the univariate linear model, and may contain, therefore, dummy regressors representing factors, polynomial or regression-spline terms, interaction regressors, and so on. For brevity, we assume that $\mathbf{X}$ is of full column-rank p; allowing for less than full rank cases would only introduce additional notation but not fundamentally change any of the results presented here.

The assumptions of the multivariate linear model concern the behavior of the errors: Let $\boldsymbol{\varepsilon}_i'$ represent the ith row of $\mathbf{E}$. Then $\boldsymbol{\varepsilon}_i' \sim \mathbf{N}_m(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a nonsingular error-covariance matrix, constant across observations; $\boldsymbol{\varepsilon}_i'$ and $\boldsymbol{\varepsilon}_j'$ are independent for $i \neq j$; and $\mathbf{X}$ is fixed or independent of $\mathbf{E}$. We can write more compactly that $\text{vec}(\mathbf{E}) \sim \mathbf{N}_{nm}(\mathbf{0}, \mathbf{I}_n \otimes \boldsymbol{\Sigma})$. Here, $\text{vec}(\mathbf{E})$ ravels the error matrix row-wise into a vector, $\mathbf{I}_n$ is the order-n identity matrix, and $\otimes$ is the Kronecker-product operator.

The maximum-likelihood estimator of $\mathbf{B}$ in the multivariate linear model is equivalent to equation-by-equation least squares for the individual responses:

$$\widehat{\mathbf{B}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

Procedures for statistical inference in the multivariate linear model, however, take account of correlations among the responses.

Paralleling the decomposition of the total sum of squares into regression and residual sums of squares in the univariate linear model, there is in the multivariate linear model a decomposition of the total *sum-of-squares-and-cross-products* (*SSP*) matrix into regression and residual SSP matrices. We have

$$\begin{aligned}\underset{(m\times m)}{\mathbf{SSP}_T} &= \mathbf{Y}'\mathbf{Y} - n\overline{\mathbf{y}}\,\overline{\mathbf{y}}' \\ &= \widehat{\mathbf{E}}'\widehat{\mathbf{E}} + \left(\widehat{\mathbf{Y}}'\widehat{\mathbf{Y}} - n\overline{\mathbf{y}}\,\overline{\mathbf{y}}'\right) \\ &= \mathbf{SSP}_R + \mathbf{SSP}_{\text{Reg}}\end{aligned}$$

where $\overline{\mathbf{y}}$ is the $(m \times 1)$ vector of means for the response variables; $\widehat{\mathbf{Y}} = \mathbf{X}\widehat{\mathbf{B}}$ is the matrix of fitted values; and $\widehat{\mathbf{E}} = \mathbf{Y} - \widehat{\mathbf{Y}}$ is the matrix of residuals.

Many hypothesis tests of interest can be formulated by taking differences in $\mathbf{SSP}_{\text{Reg}}$ (or, equivalently, $\mathbf{SSP}_R$) for nested models, although the `Anova` function in the **car** package (Fox and Weisberg, 2011), described below, calculates SSP matrices for common hypotheses more cleverly, without refitting the model. Let $\mathbf{SSP}_H$ represent the incremental SSP matrix for a hypothesis—that is, the difference between $\mathbf{SSP}_{\text{Reg}}$ for the model unrestricted by the hypothesis and $\mathbf{SSP}_{\text{Reg}}$ for the model on which the hypothesis is imposed. Multivariate tests for the hypothesis are based on the m eigenvalues λ_j of $\mathbf{SSP}_H\mathbf{SSP}_R^{-1}$ (the hypothesis SSP matrix "divided by" the residual SSP matrix), that is, the values of λ for which

$$\det(\mathbf{SSP}_H\mathbf{SSP}_R^{-1} - \lambda\mathbf{I}_m) = 0$$

The several commonly used multivariate test statistics are functions of these eigenvalues:

$$\begin{aligned}
&\text{Pillai-Bartlett Trace, } T_{PB} = \sum_{j=1}^{m} \frac{\lambda_j}{1-\lambda_j} \\
&\text{Hotelling-Lawley Trace, } T_{HL} = \sum_{j=1}^{m} \lambda_j \\
&\text{Wilks's Lambda, } \Lambda = \prod_{j=1}^{m} \frac{1}{1+\lambda_j} \\
&\text{Roy's Maximum Root, } \lambda_1
\end{aligned} \tag{1}$$

By convention, the eigenvalues of $\mathbf{SSP}_H\mathbf{SSP}_R^{-1}$ are arranged in descending order, and so λ_1 is the largest eigenvalue. The **car** package uses F approximations to the null distributions of these test statistics (see, e.g., Rao, 1973, p. 556, for Wilks's Lambda).

The tests apply generally to all linear hypotheses. Suppose that we want to test the linear hypothesis

$$H_0\colon \underset{(q\times p)}{\mathbf{L}}\ \underset{(p\times m)}{\mathbf{B}} = \underset{(q\times m)}{\mathbf{C}} \tag{2}$$

where $\mathbf{L}$ is a hypothesis matrix of full row-rank $q \leq p$, and the right-hand-side matrix $\mathbf{C}$ consists of constants, usually 0s. Then the SSP matrix for the hypothesis is

$$\mathbf{SSP}_H = \left(\widehat{\mathbf{B}}'\mathbf{L}' - \mathbf{C}'\right)\left[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}'\right]^{-1}\left(\mathbf{L}\widehat{\mathbf{B}} - \mathbf{C}\right)$$

The various test statistics are based on the $k = \min(q, m)$ nonzero eigenvalues of $\mathbf{SSP}_H\mathbf{SSP}_R^{-1}$.

When a multivariate response arises because a variable is measured on different occasions, or under different circumstances (but for the same individuals), it is also of interest to formulate hypotheses concerning comparisons among the responses. This situation, called a *repeated-measures design*, can be handled by linearly transforming the responses using a suitable "within-subjects" model matrix, for example extending the linear hypothesis in Equation 2 to

$$H_0\colon \underset{(q\times p)}{\mathbf{L}}\ \underset{(p\times m)}{\mathbf{B}}\ \underset{(m\times v)}{\mathbf{P}} = \underset{(q\times v)}{\mathbf{C}} \tag{3}$$

Here, the *response-transformation matrix* $\mathbf{P}$, assumed to be of full column-rank, provides contrasts in the responses (see, e.g., Hand and Taylor, 1987, or O'Brien and Kaiser, 1985). The SSP matrix for the hypothesis is

$$\underset{(q\times q)}{\mathbf{SSP}_H} = \left(\mathbf{P}'\widehat{\mathbf{B}}'\mathbf{L}' - \mathbf{C}'\right)\left[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}'\right]^{-1}\left(\mathbf{L}\widehat{\mathbf{B}}\mathbf{P} - \mathbf{C}\right)$$

and test statistics are based on the $k = \min(q, v)$ nonzero eigenvalues of $\mathbf{SSP}_H(\mathbf{P}'\mathbf{SSP}_R\mathbf{P})^{-1}$.

Fitting and testing multivariate linear models

Multivariate linear models are fit in R with the `lm` function. The procedure is the essence of simplicity: The left-hand side of the model formula is a matrix of responses, with each column representing a response variable and each row an observation; the right-hand side of the model formula and all other arguments to `lm` are precisely the same as for a univariate linear model (as described, e.g., in Fox and Weisberg, 2011, Chap. 4). Typically, the response matrix is composed from individual response variables via the `cbind` function. The `anova` function in the standard R distribution is capable of handling multivariate linear models (see Dalgaard, 2007), but the `Anova` and `linearHypothesis` functions in the **car** package may also be employed. We briefly demonstrate the use of these functions in this section.

Figure 1: Three species of irises in the Anderson/Fisher data set: setosa (left), versicolor (center), and virginica (right). *Source*: The photographs are respectively by Radomil Binek, Danielle Langlois, and Frank Mayfield, and are distributed under the Creative Commons Attribution-Share Alike 3.0 Unported license (first and second images) or 2.0 Creative Commons Attribution-Share Alike Generic license (third image); they were obtained from the Wikimedia Commons.

`Anova` and `linearHypothesis` are generic functions with methods for many common classes of statistical models with linear predictors. In addition to multivariate linear models, these classes include linear models fit by `lm` or `aov`; generalized linear models fit by `glm`; mixed-effects models fit by `lmer` or `glmer` in the **lme4** package (Bates et al., 2012) or `lme` in the **nlme** package (Pinheiro et al., 2012); survival regression models fit by `coxph` or `survreg` in the **survival** package (Therneau, 2012); multinomial-response models fit by `multinom` in the **nnet** package (Venables and Ripley, 2002); ordinal regression models fit by `polr` in the **MASS** package (Venables and Ripley, 2002); and generalized linear models fit to complex-survey data via `svyglm` in the **survey** package (Lumley, 2004). There is also a generic method that will work with many models for which there are `coef` and `vcov` methods. The `Anova` and `linearHypothesis` methods for `"mlm"` objects are special, however, in that they handle multiple response variables and make provision for designs on repeated measures, discussed in the next section.

To illustrate multivariate linear models, we will use data collected by Anderson (1935) on three species of irises in the Gaspé Peninsula of Québec, Canada. The data are of historical interest in statistics, because they were employed by R. A. Fisher (1936) to introduce the method of discriminant analysis. The data frame `iris` is part of the standard R distribution, and we load the **car** package now for the `some` function, which randomly samples the rows of a data set. We rename the variables in the `iris` data to make listings more compact:

```
> names(iris)

[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"

> names(iris) <- c("SL", "SW", "PL", "PW", "SPP")
> library(car)
> some(iris, 3) # 3 random rows

     SL  SW  PL  PW        SPP
44  5.0 3.5 1.6 0.6     setosa
61  5.0 2.0 3.5 1.0 versicolor
118 7.7 3.8 6.7 2.2  virginica
```

The first four variables in the data set represent measurements (in cm) of parts of the flowers, while the final variable specifies the species of iris. (Sepals are the green leaves that comprise the calyx of the plant, which encloses the flower.) Photographs of examples of the three species of irises—setosa, versicolor, and virginica—appear in Figure 1. Figure 2 is a scatterplot matrix of the four measurements classified by species, showing within-species 50 and 95% concentration ellipses (see Fox and Weisberg, 2011, Sec. 4.3.8); Figure 3 shows boxplots for each of the responses by species. These graphs are produced by the `scatterplotMatrix` and `Boxplot` functions in the **car** package (see Fox and Weisberg, 2011, Sec. 3.2.2 and 3.3.2). As the photographs suggest, the scatterplot matrix and boxplots for the measurements reveal that versicolor and virginica are more similar to each other than either is to setosa. Further, the ellipses in the scatterplot matrix suggest that the assumption of constant within-group covariance matrices is problematic: While the shapes and sizes of the concentration ellipses for versicolor and virginica are reasonably similar, the shapes and sizes of the ellipses for setosa are different from the other two.

We proceed nevertheless to fit a multivariate one-way ANOVA model to the iris data:

```
> mod.iris <- lm(cbind(SL, SW, PL, PW) ~ SPP, data=iris)
> class(mod.iris)
```

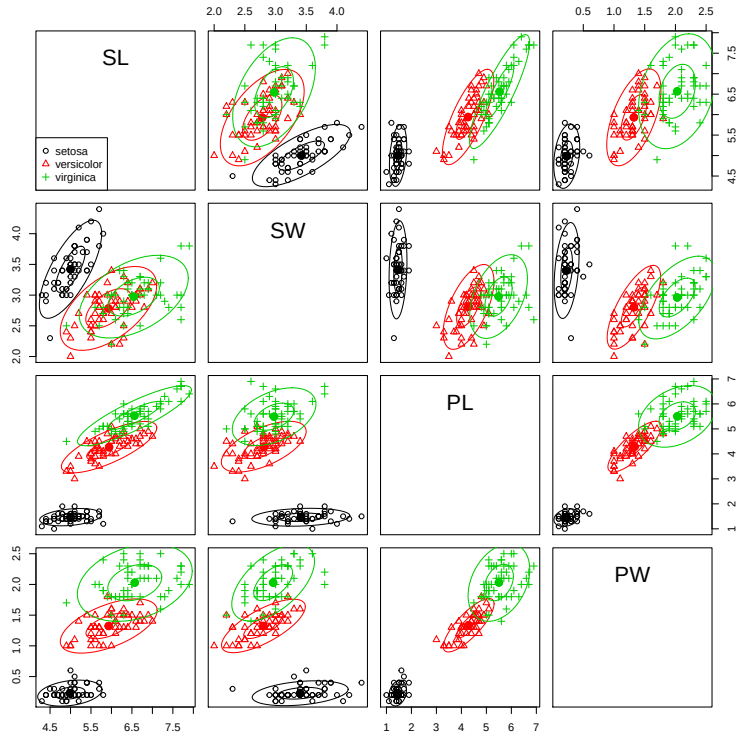


Figure 2: Scatterplot matrix for the Anderson/Fisher iris data, showing within-species 50 and 95% concentration ellipses.

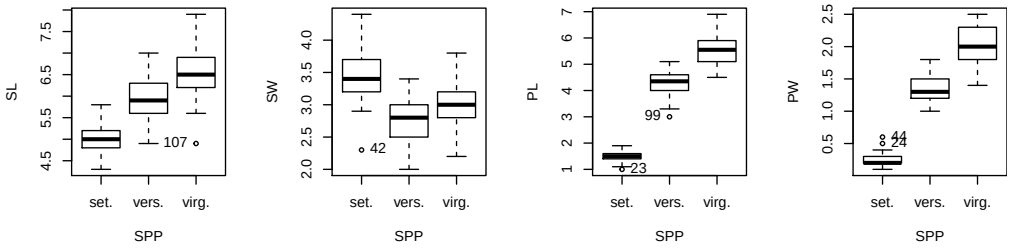


Figure 3: Boxplots for the response variables in the iris data set classified by species.

```
[1] "mlm" "lm"
```

The `lm` function returns an S3 object of class `"mlm"` inheriting from class `"lm"`. The printed representation of the object (not shown) simply displays the estimated regression coefficients for each response, and the model summary (also not shown) is the same as we would obtain by performing separate least-squares regressions for the four responses.

We use the `Anova` function in the **car** package to test the null hypothesis that the four response means are identical across the three species of irises:

```
> manova.iris <- Anova(mod.iris)
> manova.iris

Type II MANOVA Tests: Pillai test statistic
    Df test stat approx F num Df den Df Pr(>F)
SPP  2      1.19     53.5      8    290 <2e-16

> class(manova.iris)

[1] "Anova.mlm"

> summary(manova.iris)
```

```
Type II MANOVA Tests:

Sum of squares and products for error:
       SL     SW     PL    PW
SL 38.956 13.630 24.625 5.645
SW 13.630 16.962  8.121 4.808
PL 24.625  8.121 27.223 6.272
PW  5.645  4.808  6.272 6.157

----------------------------------------

Term: SPP

Sum of squares and products for the hypothesis:
       SL     SW     PL     PW
SL  63.21 -19.95 165.25  71.28
SW -19.95  11.34 -57.24 -22.93
PL 165.25 -57.24 437.10 186.77
PW  71.28 -22.93 186.77  80.41

Multivariate Tests: SPP
                 Df test stat approx F num Df den Df Pr(>F)
Pillai            2      1.19     53.5      8    290 <2e-16
Wilks             2      0.02    199.1      8    288 <2e-16
Hotelling-Lawley  2     32.48    580.5      8    286 <2e-16
Roy               2     32.19   1167.0      4    145 <2e-16
```

The Anova function returns an object of class "Anova.mlm" which, when printed, produces a MANOVA table, by default reporting Pillai's test statistic;[1] summarizing the object produces a more complete report. Because there is only one term (beyond the regression constant) on the right-hand side of the model, in this example the "type-II" test produced by default by Anova is the same as the sequential ("type-I") test produced by the standard R anova function (output not shown):

```
> anova(mod.iris)
```

The null hypothesis is soundly rejected.

The object returned by Anova may also be used in further computations, for example, for displays such as *hypothesis-error (HE) plots* (Friendly, 2007; Fox et al., 2009; Friendly, 2010), as we illustrate below.

The linearHypothesis function in the **car** package may be used to test more specific hypotheses about the parameters in the multivariate linear model. For example, to test for differences between setosa and the average of versicolor and virginica, and for differences between versicolor and virginica:

```
> linearHypothesis(mod.iris, "0.5*SPPversicolor + 0.5*SPPvirginica")

. . .
Multivariate Tests:
                 Df test stat approx F num Df den Df Pr(>F)
Pillai            1     0.967     1064      4    144 <2e-16
Wilks             1     0.033     1064      4    144 <2e-16
Hotelling-Lawley  1    29.552     1064      4    144 <2e-16
Roy               1    29.552     1064      4    144 <2e-16

> linearHypothesis(mod.iris, "SPPversicolor = SPPvirginica")

. . .
Multivariate Tests:
                 Df test stat approx F num Df den Df Pr(>F)
Pillai            1    0.7452    105.3      4    144 <2e-16
Wilks             1    0.2548    105.3      4    144 <2e-16
Hotelling-Lawley  1    2.9254    105.3      4    144 <2e-16
Roy               1    2.9254    105.3      4    144 <2e-16
```

[1] The Manova function in the **car** package may be used as a synonym for Anova applied to a multivariate linear model. The computation of the various multivariate test statistics is performed via unexported functions from the standard R **stats** package, such as stats:::Pillai.

Here and elsewhere in this paper, we use widely separated ellipses (. . .) to indicate abbreviated R output.

Setting the argument `verbose=TRUE` to `linearHypothesis` (not given here to conserve space) shows in addition the hypothesis matrix **L** and right-hand-side matrix **C** for the linear hypothesis in Equation 2 (page 40). In this case, all of the multivariate test statistics are equivalent and therefore translate into identical *F*-statistics. Both focussed null hypotheses are easily rejected, but the evidence for differences between setosa and the other two iris species is much stronger than for differences between versicolor and virginica. Testing that `"0.5*SPPversicolor + 0.5*SPPvirginica"` is **0** tests that the average of the mean vectors for these two species is equal to the mean vector for setosa, because the latter is the baseline category for the species dummy regressors.

An alternative, equivalent, and in a sense more direct, approach is to fit the model with custom contrasts for the three species of irises, followed up by a test for each contrast:

```
> C <- matrix(c(1, -0.5, -0.5, 0, 1, -1), 3, 2)
> colnames(C) <- c("S:VV", "V:V")
> rownames(C) <- unique(iris$SPP)
> contrasts(iris$SPP) <- C
> contrasts(iris$SPP)

           S:VV V:V
setosa      1.0   0
versicolor -0.5   1
virginica  -0.5  -1

> mod.iris.2 <- update(mod.iris)
> coef(mod.iris.2)

                 SL      SW     PL      PW
(Intercept)  5.8433  3.0573  3.758  1.1993
SPPS:VV     -0.8373  0.3707 -2.296 -0.9533
SPPV:V      -0.3260 -0.1020 -0.646 -0.3500

> linearHypothesis(mod.iris.2, c(0, 1, 0)) # setosa vs. versicolor & virginica

. . .
Multivariate Tests:
                 Df test stat approx F num Df den Df Pr(>F)
Pillai            1     0.967     1064      4    144 <2e-16
Wilks             1     0.033     1064      4    144 <2e-16
Hotelling-Lawley  1    29.552     1064      4    144 <2e-16
Roy               1    29.552     1064      4    144 <2e-16

> linearHypothesis(mod.iris.2, c(0, 0, 1)) # versicolor vs. virginica

. . .
Multivariate Tests:
                 Df test stat approx F num Df den Df Pr(>F)
Pillai            1    0.7452    105.3      4    144 <2e-16
Wilks             1    0.2548    105.3      4    144 <2e-16
Hotelling-Lawley  1    2.9254    105.3      4    144 <2e-16
Roy               1    2.9254    105.3      4    144 <2e-16
```

We note here briefly that the **heplots** package (Friendly, 2007; Fox et al., 2009) provides informative visualizations in 2D and 3D HE plots of multivariate hypothesis tests and `"Anova.mlm"` objects based on Eqn. 2. These plots show direct visual representations of the $\mathbf{SSP}_H$ and $\mathbf{SSP}_E$ matrices as (possibly degenerate) ellipses and ellipsoids.

Using the default *significance scaling*, HE plots have the property that the $\mathbf{SSP}_H$ ellipsoid extends outside the $\mathbf{SSP}_E$ ellipsoid *if and only if* the corresponding multivariate hypothesis test is rejected by Roy's maximum root test at a given α level. See Friendly (2007) and Fox et al. (2009) for details of these methods, and Friendly (2010) for analogous plots for repeated measure designs.

To illustrate, Figure 4 shows the 2D HE plot of the two sepal variables for the overall test of species, together with the tests of the contrasts among species described above. The $\mathbf{SSP}_H$ matrices for the contrasts have rank 1, so their ellipses plot as lines. All three $\mathbf{SSP}_H$ ellipses extend far outside the $\mathbf{SSP}_E$ ellipse, indicating that all tests are highly significant.

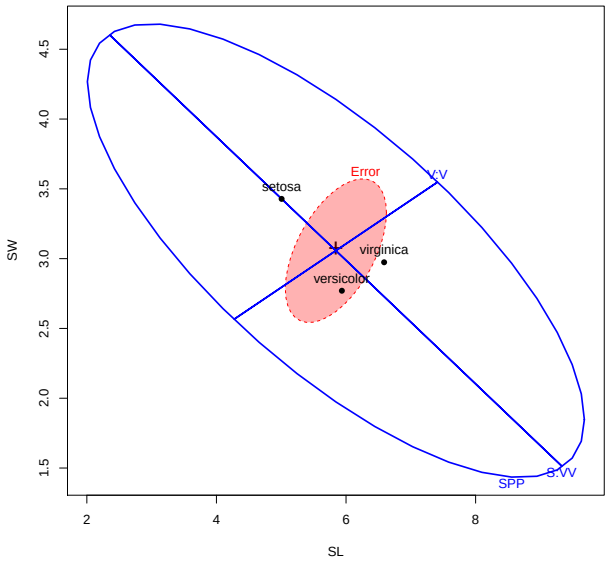


Figure 4: HE plot for the multivariate test of species in the iris data, $\alpha = 0.05$, shown for the sepal length and sepal width response variables. Also shown are the multivariate `linearHypothesis` tests for two contrasts among species. The shaded red ellipse is the error ellipse, and the hypothesis ellipses (including the two lines) are blue.

```
> library(heplots)
> hyp <- list("V:V"="SPPV:V", "S:VV"="SPPS:VV")
> heplot(mod.iris.2, hypotheses=hyp, fill=c(TRUE, FALSE), col=c("red", "blue"))
```

Finally, we can code the response-transformation matrix **P** in Equation 3 (page 40) to compute linear combinations of the responses, either via the `imatrix` argument to `Anova` (which takes a list of matrices) or the `P` argument to `linearHypothesis` (which takes a matrix). We illustrate trivially with a univariate ANOVA for the first response variable, sepal length, extracted from the multivariate linear model for all four responses:

```
> Anova(mod.iris, imatrix=list(Sepal.Length=matrix(c(1, 0, 0, 0))))

Type II Repeated Measures MANOVA Tests: Pillai test statistic
                Df test stat approx F num Df den Df Pr(>F)
Sepal.Length     1     0.992    19327      1    147 <2e-16
SPP:Sepal.Length 2     0.619      119      2    147 <2e-16
```

The univariate ANOVA for sepal length by species appears in the second line of the MANOVA table produced by `Anova`. Similarly, using `linearHypothesis`,

```
> linearHypothesis(mod.iris, c("SPPversicolor = 0", "SPPvirginica = 0"),
+   P=matrix(c(1, 0, 0, 0))) # equivalent

. . .
Multivariate Tests:
                Df test stat approx F num Df den Df Pr(>F)
Pillai           2    0.6187    119.3      2    147 <2e-16
Wilks            2    0.3813    119.3      2    147 <2e-16
Hotelling-Lawley 2    1.6226    119.3      2    147 <2e-16
Roy              2    1.6226    119.3      2    147 <2e-16
```

In this case, the `P` matrix is a single column picking out the first response. We verify that we get the same F-test from a univariate ANOVA for `Sepal.Length`:

```
> Anova(lm(SL ~ SPP, data=iris))

Anova Table (Type II tests)
```

```
Response: SL
          Sum Sq  Df F value Pr(>F)
SPP         63.2   2     119 <2e-16
Residuals   39.0 147
```

Contrasts of the responses occur more naturally in the context of repeated-measures data, which we discuss in the following section.

Handling repeated measures

Repeated-measures data arise when multivariate responses represent the same individuals measured on a response variable (or variables) on different occasions or under different circumstances. There may be a more or less complex design on the repeated measures. The simplest case is that of a single repeated-measures or *within-subjects* factor, where the former term often is applied to data collected over time and the latter when the responses represent different experimental conditions or treatments. There may, however, be two or more within-subjects factors, as is the case, for example, when each subject is observed under different conditions on each of several occasions. The terms "repeated measures" and "within-subjects factors" are common in disciplines, such as psychology, where the units of observation are individuals, but these designs are essentially the same as so-called "split-plot" designs in agriculture, where plots of land are each divided into sub-plots, which are subjected to different experimental treatments, such as differing varieties of a crop or differing levels of fertilizer.

Repeated-measures designs can be handled in R with the standard `anova` function, as described by Dalgaard (2007), but it is considerably simpler to get common tests from the functions `Anova` and `linearHypothesis` in the **car** package, as we explain in this section. The general procedure is first to fit a multivariate linear model with all of the repeated measures as responses; then an artificial data frame is created in which each of the repeated measures is a row and in which the columns represent the repeated-measures factor or factors; finally, as we explain below, the `Anova` or `linearHypothesis` function is called, using the `idata` and `idesign` arguments (and optionally the `icontrasts` argument)—or alternatively the `imatrix` argument to `Anova` or `P` argument to `linearHypothesis`—to specify the intra-subject design.

To illustrate, we use data reported by O'Brien and Kaiser (1985), in what they (justifiably) bill as "an extensive primer" for the MANOVA approach to repeated-measures designs. Although the data are apparently not real, they are contrived cleverly to illustrate the computations for repeated-measures MANOVA, and we use the data for this reason, as well as to permit comparison of our results to those in an influential published source. The data set `OBrienKaiser` is provided by the **car** package:

```
> some(OBrienKaiser, 4)

   treatment gender pre.1 pre.2 pre.3 pre.4 pre.5 post.1 post.2 post.3 post.4
11         B      M     3     3     4     2     3      5      4      7      5
12         B      M     6     7     8     6     3      9     10     11      9
14         B      F     2     2     3     1     2      5      6      7      5
16         B      F     4     5     7     5     4      7      7      8      6
   post.5 fup.1 fup.2 fup.3 fup.4 fup.5
11      4     5     6     8     6     5
12      6     8     7    10     8     7
14      2     6     7     8     6     3
16      7     7     8    10     8     7

> contrasts(OBrienKaiser$treatment)

        [,1] [,2]
control   -2    0
A          1   -1
B          1    1

> contrasts(OBrienKaiser$gender)

  [,1]
F    1
M   -1

> xtabs(~ treatment + gender, data=OBrienKaiser)
```

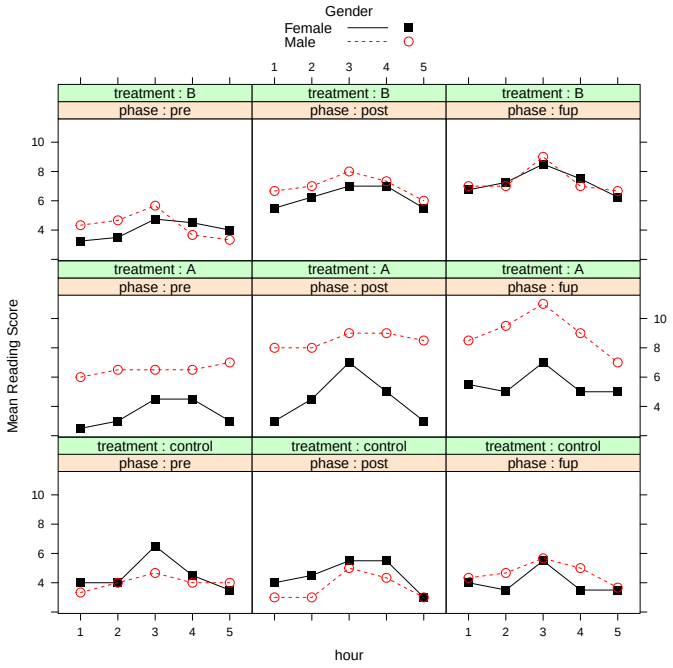


Figure 5: Mean reading score by gender, treatment, phase, and hour, for the O'Brien-Kaiser data.

```
          gender
treatment F M
  control 2 3
  A       2 2
  B       4 3
```

There are two between-subjects factors in the O'Brien-Kaiser data: `gender`, with levels `F` and `M`; and `treatment`, with levels `A`, `B`, and `control`. Both of these variables have predefined contrasts, with $-1, 1$ coding for `gender` and custom contrasts for `treatment`. In the latter case, the first contrast is for the `control` group vs. the average of the experimental groups, and the second contrast is for treatment `A` vs. treatment `B`. We have defined these contrasts, which are orthogonal in the row-basis of the between-subjects design, to reproduce the type-III tests that are reported in the original source.

The frequency table for `treatment` by `gender` reveals that the data are mildly unbalanced. We will imagine that the treatments `A` and `B` represent different innovative methods of teaching reading to learning-disabled students, and that the `control` treatment represents a standard method.

The 15 response variables in the data set represent two crossed within-subjects factors: *phase*, with three levels for the *pretest*, *post-test*, and *follow-up* phases of the study; and *hour*, representing five successive hours, at which measurements of reading comprehension are taken within each phase. We define the "data" for the within-subjects design as follows:

```
> phase <- factor(rep(c("pretest", "posttest", "followup"), each=5),
+   levels=c("pretest", "posttest", "followup"))
> hour <- ordered(rep(1:5, 3))
> idata <- data.frame(phase, hour)
> idata

      phase hour
1   pretest    1
2   pretest    2
3   pretest    3
. . .
14 followup    4
15 followup    5
```

Mean reading comprehension is graphed by hour, phase, treatment, and gender in Figure 5. It appears as if reading improves across phases in the two experimental treatments but not in the control group (suggesting a possible treatment-by-phase interaction); that there is a possibly quadratic relationship of reading to hour within each phase, with an initial rise and then decline, perhaps

representing fatigue (suggesting an hour main effect); and that males and females respond similarly in the control and B treatment groups, but that males do better than females in the A treatment group (suggesting a possible gender-by-treatment interaction).

We next fit a multivariate linear model to the data, treating the repeated measures as responses, and with the between-subject factors `treatment` and `gender` (and their interaction) appearing on the right-hand side of the model formula:

```
> mod.ok <- lm(cbind(pre.1, pre.2, pre.3, pre.4, pre.5,
+                    post.1, post.2, post.3, post.4, post.5,
+                    fup.1, fup.2, fup.3, fup.4, fup.5)
+                ~  treatment*gender, data=OBrienKaiser)
```

We then compute the repeated-measures MANOVA using the `Anova` function in the following manner:

```
> av.ok <- Anova(mod.ok, idata=idata, idesign=~phase*hour, type=3)
> av.ok

Type III Repeated Measures MANOVA Tests: Pillai test statistic
                          Df test stat approx F num Df den Df  Pr(>F)
(Intercept)                1     0.967    296.4      1     10 9.2e-09
treatment                  2     0.441      3.9      2     10 0.05471
gender                     1     0.268      3.7      1     10 0.08480
treatment:gender           2     0.364      2.9      2     10 0.10447
phase                      1     0.814     19.6      2      9 0.00052
treatment:phase            2     0.696      2.7      4     20 0.06211
gender:phase               1     0.066      0.3      2      9 0.73497
treatment:gender:phase     2     0.311      0.9      4     20 0.47215
hour                       1     0.933     24.3      4      7 0.00033
treatment:hour             2     0.316      0.4      8     16 0.91833
gender:hour                1     0.339      0.9      4      7 0.51298
treatment:gender:hour      2     0.570      0.8      8     16 0.61319
phase:hour                 1     0.560      0.5      8      3 0.82027
treatment:phase:hour       2     0.662      0.2     16      8 0.99155
gender:phase:hour          1     0.712      0.9      8      3 0.58949
treatment:gender:phase:hour 2    0.793      0.3     16      8 0.97237
```

- Following O'Brien and Kaiser (1985), we report type-III tests (partial tests violating marginality), by specifying the argument `type=3`. Although, as in univariate models, we generally prefer type-II tests (see Fox and Weisberg, 2011, Sec. 4.4.4, and Fox, 2008, Sec. 8.2), we wanted to preserve comparability with the original source. Type-III tests are computed correctly because the contrasts employed for `treatment` and `gender`, and hence their interaction, are orthogonal in the row-basis of the between-subjects design. We invite the reader to compare these results with the default type-II tests.
- When, as here, the `idata` and `idesign` arguments are specified, `Anova` automatically constructs orthogonal contrasts for different terms in the within-subjects design, using `contr.sum` for a factor such as `phase` and `contr.poly` (orthogonal polynomial contrasts) for an ordered factor such as `hour`. Alternatively, the user can assign contrasts to the columns of the intra-subject data, either directly or via the `icontrasts` argument to `Anova`. In any event, `Anova` checks that the within-subjects contrast coding for different terms is orthogonal and reports an error when it is not.
- By default, Pillai's test statistic is displayed; we invite the reader to examine the other three multivariate test statistics. Much more detail of the tests is provided by `summary(av.ok)` (not shown).
- The results show that the anticipated `hour` effect is statistically significant, but the `treatment` $\times$ `phase` and `treatment` $\times$ `gender` interactions are not quite significant. There is, however, a statistically significant `phase` main effect. Of course, we should not over-interpret these results, partly because the data set is small and partly because it is contrived.

Univariate ANOVA for repeated measures

A traditional univariate approach to repeated-measures (or split-plot) designs (see, e.g., Winer, 1971, Chap. 7) computes an analysis of variance employing a "mixed-effects" model in which subjects generate random effects. This approach makes stronger assumptions about the structure of the data than the MANOVA approach described above, in particular stipulating that the covariance matrices

for the repeated measures transformed by the within-subjects design (within combinations of between-subjects factors) are *spherical*—that is, the transformed repeated measures for each within-subjects test are uncorrelated and have the same variance, and this variance is constant across cells of the between-subjects design. A sufficient (but not necessary) condition for sphericity of the errors is that the covariance matrix $\mathbf{\Sigma}$ of the repeated measures is *compound-symmetric*, with equal diagonal entries (representing constant variance for the repeated measures) and equal off-diagonal elements (implying, together with constant variance, that the repeated measures have a constant correlation).

By default, when an intra-subject design is specified, summarizing the object produced by `Anova` reports both MANOVA and univariate tests. Along with the traditional univariate tests, the summary reports tests for sphericity (Mauchly, 1940) and two corrections for non-sphericity of the univariate test statistics for within-subjects terms: the Greenhouse-Geisser correction (Greenhouse and Geisser, 1959) and the Huynh-Feldt correction (Huynh and Feldt, 1976). We illustrate for the O'Brien-Kaiser data, suppressing the output for brevity; we invite the reader to reproduce this analysis:

```
> summary(av.ok, multivariate=FALSE)
```

There are statistically significant departures from sphericity for *F*-tests involving `hour`; the results for the univariate ANOVA are not terribly different from those of the MANOVA reported above, except that now the `treatment` × `phase` interaction is statistically significant.

Using `linearHypothesis` with repeated-measures designs

As for simpler multivariate linear models (discussed previously in this paper), the `linearHypothesis` function can be used to test more focused hypotheses about the parameters of repeated-measures models, including for within-subjects terms.

As a preliminary example, to reproduce the test for the main effect of `hour`, we can use the `idata`, `idesign`, and `iterms` arguments in a call to `linearHypothesis`:

```
> linearHypothesis(mod.ok, "(Intercept) = 0", idata=idata,
+   idesign=~phase*hour, iterms="hour")

 Response transformation matrix:
        hour.L  hour.Q     hour.C  hour^4
pre.1  -0.6325  0.5345 -3.162e-01  0.1195
pre.2  -0.3162 -0.2673  6.325e-01 -0.4781
. . .
fup.5   0.6325  0.5345  3.162e-01  0.1195

. . .

Multivariate Tests:
                 Df test stat approx F num Df den Df   Pr(>F)
Pillai            1     0.933    24.32      4      7 0.000334
Wilks             1     0.067    24.32      4      7 0.000334
Hotelling-Lawley  1    13.894    24.32      4      7 0.000334
Roy               1    13.894    24.32      4      7 0.000334
```

Because `hour` is a within-subjects factor, we test its main effect as the regression intercept in the between-subjects model, using a response-transformation matrix for the `hour` contrasts.

Alternatively and equivalently, we can generate the response-transformation matrix `P` for the hypothesis directly:

```
> Hour <- model.matrix(~ hour, data=idata)
> dim(Hour)

[1] 15  5

> head(Hour, 5)

  (Intercept)  hour.L  hour.Q     hour.C  hour^4
1           1 -0.6325  0.5345 -3.162e-01  0.1195
2           1 -0.3162 -0.2673  6.325e-01 -0.4781
3           1  0.0000 -0.5345 -4.096e-16  0.7171
4           1  0.3162 -0.2673 -6.325e-01 -0.4781
5           1  0.6325  0.5345  3.162e-01  0.1195
```

```
> linearHypothesis(mod.ok, "(Intercept) = 0", P=Hour[ , c(2:5)])

 Response transformation matrix:
       hour.L  hour.Q     hour.C  hour^4
pre.1  -0.6325  0.5345 -3.162e-01  0.1195
pre.2  -0.3162 -0.2673  6.325e-01 -0.4781
. . .
fup.5   0.6325  0.5345  3.162e-01  0.1195

Sum of squares and products for the hypothesis:
        hour.L   hour.Q   hour.C    hour^4
hour.L  0.01034    1.556   0.3672   -0.8244
hour.Q  1.55625  234.118  55.2469 -124.0137
hour.C  0.36724   55.247  13.0371  -29.2646
hour^4 -0.82435 -124.014 -29.2646   65.6907

. . .

Multivariate Tests:
                 Df test stat approx F num Df den Df   Pr(>F)
Pillai            1     0.933    24.32      4      7 0.000334
Wilks             1     0.067    24.32      4      7 0.000334
Hotelling-Lawley  1    13.894    24.32      4      7 0.000334
Roy               1    13.894    24.32      4      7 0.000334
```

As mentioned, this test simply duplicates part of the output from Anova, but suppose that we want to test the individual polynomial components of the hour main effect:

```
> linearHypothesis(mod.ok, "(Intercept) = 0", P=Hour[ , 2, drop=FALSE]) # linear

 Response transformation matrix:
       hour.L
pre.1  -0.6325
pre.2  -0.3162
. . .
fup.5   0.6325

. . .

Multivariate Tests:
                 Df test stat approx F num Df den Df Pr(>F)
Pillai            1    0.0001 0.001153      1     10  0.974
Wilks             1    0.9999 0.001153      1     10  0.974
Hotelling-Lawley  1    0.0001 0.001153      1     10  0.974
Roy               1    0.0001 0.001153      1     10  0.974

> linearHypothesis(mod.ok, "(Intercept) = 0", P=Hour[ , 3, drop=FALSE]) # quadratic

 Response transformation matrix:
       hour.Q
pre.1   0.5345
pre.2  -0.2673
. . .
fup.5   0.5345

. . .

Multivariate Tests:
                 Df test stat approx F num Df den Df    Pr(>F)
Pillai            1     0.834    50.19      1     10 0.0000336
Wilks             1     0.166    50.19      1     10 0.0000336
Hotelling-Lawley  1     5.019    50.19      1     10 0.0000336
Roy               1     5.019    50.19      1     10 0.0000336

> linearHypothesis(mod.ok, "(Intercept) = 0", P=Hour[ , c(2, 4:5)]) # all non-quadratic
```

```
 Response transformation matrix:
       hour.L     hour.C  hour^4
pre.1  -0.6325 -3.162e-01  0.1195
pre.2  -0.3162  6.325e-01 -0.4781
. . .
fup.5   0.6325  3.162e-01  0.1195

. . .

Multivariate Tests:
                 Df test stat approx F num Df den Df   Pr(>F)
Pillai            1     0.896    23.05      3      8 0.000272
Wilks             1     0.104    23.05      3      8 0.000272
Hotelling-Lawley  1     8.644    23.05      3      8 0.000272
Roy               1     8.644    23.05      3      8 0.000272
```

The hour main effect is more complex, therefore, than a simple quadratic trend.

Conclusions

In contrast to the standard R anova function, the Anova and linearHypothesis functions in the **car** package make it relatively simple to compute hypothesis tests that are typically used in applications of multivariate linear models, including repeated-measures data. Although similar facilities for multivariate analysis of variance and repeated measures are provided by traditional statistical packages such as SAS and SPSS, we believe that the printed output from Anova and linearHypothesis is more readable, producing compact standard output and providing details when one wants them. These functions also return objects containing information—for example, SSP and response-transformation matrices—that may be used for further computations and in graphical displays, such as HE plots.

Acknowledgments

The work reported in this paper was partly supported by grants to John Fox from the Social Sciences and Humanities Research Council of Canada and from the McMaster University Senator William McMaster Chair in Social Statistics.

Bibliography

E. Anderson. The irises of the Gaspé Peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935. [p41]

D. Bates, M. Maechler, and B. Bolker. *lme4: Linear Mixed-Effects Models using S4 Classes*, 2012. R package version 0.999999-0. [p41]

P. Dalgaard. New functions for multivariate analysis. *R News*, 7(2):2–7, 2007. [p40, 46]

R. A. Fisher. The use of multiple measurements in taxonomic problems. *The Annals of Eugenics*, 7, Part II:179–188, 1936. [p41]

J. Fox. *Applied Regression Analysis and Generalized Linear Models*. Sage, Thousand Oaks, CA, second edition, 2008. [p39, 48]

J. Fox and S. Weisberg. *An R Companion to Applied Regression*. Sage, Thousand Oaks, CA, second edition, 2011. [p40, 41, 48]

J. Fox, M. Friendly, and G. Monette. Visualizing hypothesis tests in multivariate linear models: The heplots package for R. *Computational Statistics*, 24:233–246, 2009. [p43, 44]

M. Friendly. HE plots for multivariate linear models. *Journal of Computational and Graphical Statistics*, 16:421–444, 2007. [p43, 44]

M. Friendly. HE plots for repeated measures designs. *Journal of Statistical Software*, 37(4):1–40, 2010. [p43, 44]

S. W. Greenhouse and S. Geisser. On methods in the analysis of profile data. *Psychometrika*, 24:95–112, 1959. [p49]

D. J. Hand and C. C. Taylor. *Multivariate Analysis of Variance and Repeated Measures: A Practical Approach for Behavioural Scientists*. Chapman and Hall, London, 1987. [p39, 40]

H. Huynh and L. S. Feldt. Estimation of the Box correction for degrees of freedom from sample data in randomized block and split-plot designs. *Journal of Educational Statistics*, 1:69–82, 1976. [p49]

T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(1):1–19, 2004. [p41]

J. W. Mauchly. Significance test for sphericity of a normal n-variate distribution. *The Annals of Mathematical Statistics*, 11:204–209, 1940. [p49]

D. F. Morrison. *Multivariate Statistical Methods*. Duxbury, Belmont CA, fourth edition, 2005. [p39]

R. G. O'Brien and M. K. Kaiser. MANOVA method for analyzing repeated measures designs: An extensive primer. *Psychological Bulletin*, 97:316–333, 1985. [p39, 40, 46, 48]

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2012. R package version 3.1-105. [p41]

C. R. Rao. *Linear Statistical Inference and Its Applications*. Wiley, New York, second edition, 1973. [p39, 40]

T. Therneau. *A Package for Survival Analysis in S*, 2012. R package version 2.36-14. [p41]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. [p41]

B. J. Winer. *Statistical Principles in Experimental Design*. McGraw-Hill, New York, second edition, 1971. [p39, 48]

John Fox
Department of Sociology
McMaster University
Canada
jfox@mcmaster.ca

Michael Friendly
Psychology Department
York University
Canada
friendly@yorku.ca

Sanford Weisberg
School of Statistics
University of Minnesota
USA
sandy@umn.edu

osmar: OpenStreetMap and R

by Manuel J. A. Eugster and Thomas Schlesinger

Abstract OpenStreetMap provides freely accessible and editable geographic data. The **osmar** package smoothly integrates the OpenStreetMap project into the R ecosystem. The **osmar** package provides infrastructure to access OpenStreetMap data from different sources, to enable working with the OSM data in the familiar R idiom, and to convert the data into objects based on classes provided by existing R packages. This paper explains the package's concept and shows how to use it. As an application we present a simple navigation device.

Introduction

"*OpenStreetMap creates and provides free geographic data such as street maps to anyone who wants them*" announces the OpenStreetMap wiki main page (OSM Foundation, 2011) – and we think R users want free geographic data. Therefore, the add-on package **osmar** (Schlesinger and Eugster, 2012) provides extensible infrastructure for integrating the OpenStreetMap project (OSM) into the R project.

The aim of the OpenStreetMap project is to create a free editable map of the world. The project maintains a database of geographic elements (nodes, ways and relations) and features (such as streets, buildings and landmarks). These data are collected and provided by volunteers using GPS devices, aerial imagery, and local knowledge. The most prominent application is the rendering of the geographic data and features into raster images (for example, for the OSM map on the website). However, the project also provides an application programming interface (API) for fetching raw data from and saving to the OSM database.

The OpenStreetMap project provides data in the OSM XML format, which consists of three basic elements:

Node: The basic element. It consists of the attributes latitude and longitude.

Way: An ordered interconnection of nodes to describe a linear feature (e.g., a street). Areas (e.g., buildings) are represented as closed ways.

Relation: A grouping of elements (nodes, ways, and relations), which are somehow geographically related (e.g., bus and cycle routes).

Each element has further *attributes* like the element ID (unique within the corresponding element group) and timestamp. Furthermore, each element may have an arbitrary number of *tags* (key-value pairs) which describe the element. Ways and relations, in addition, have *references* to their members' IDs.

In order to access the data, OSM provides an application programming interface (API) over the hypertext transfer protocol (HTTP) for getting raw data from and putting it to the OSM database. The *main API* (currently in version 0.6) has calls to get elements (and all other elements referenced by it) by, among other things, their ID and a bounding box. However, the requests are limited (e.g., currently only an area of 0.25 square degrees can be queried). An (unlimited) alternative is provided by *planet files*. These are compressed OSM XML files containing different OSM database extracts (e.g., the entire world or an individual country or area). Planet files can be downloaded from the OSM wiki and processed using the command-line Java tool Osmosis (Henderson, 2011).

For a complete introduction into the OSM project, the OSM API, and the OSM XML file format we refer to the project's wiki available at `http://wiki.openstreetmap.org/`.

The aim of the package **osmar** is to provide extensible infrastructure to get and to represent the above described OSM data within R, to enable working with the OSM data in the familiar R idiom, and to convert the OSM data to objects based on classes provided by other packages. Figure 1 visualizes the package's concept. This is a different idea than existing packages like **OpenStreetMap** (Fellows, 2012), **RgoogleMaps** (Loecher, 2012), and **ggmap** (Kahle and Wickham, 2012) follow. Whereas these packages provide access to the already rendered data (i.e., raster images), **osmar** enables the usage of the raw OSM data.

In the following section we present the package's implementation and usage. Note that we try to increase readability by only showing the relevant arguments of plot statements. We refer to the "navigator" demo in the **osmar** package for the actual plot statements.

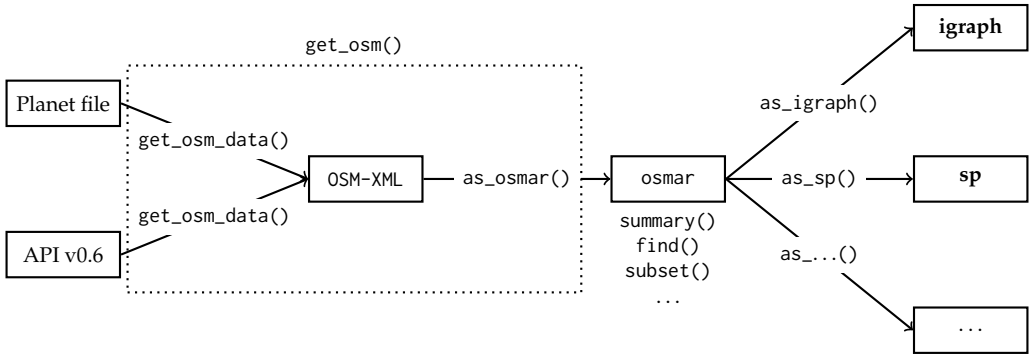


Figure 1: Schematic description of the **osmar** concept.

Getting the data

We begin with defining the data source. Currently two sources, HTTP-API and planet files, are supported. In this section we use the API of the OSM project with the default URL `url = http://api.openstreetmap.org/api/0.6/`:

```
> library("osmar")

Loading required package: XML
Loading required package: RCurl
Loading required package: bitops
Loading required package: gtools
Loading required package: geosphere
Loading required package: sp

Attaching package: 'osmar'

The following object(s) are masked
from 'package:utils':

    find

> src <- osmsource_api()
```

We can retrieve elements by using the IDs of the elements. The IDs in these examples have been extracted by hand from the OpenStreetMap website (via its export functionality). For example, one node:

```
> get_osm(node(18961430), source = src)

osmar object
1 nodes, 0 ways, 0 relations
```

Or, one way with the way-related data only or with the data for all referenced elements (nodes and relations):

```
> get_osm(way(3810479), source = src)

osmar object
0 nodes, 1 ways, 0 relations

> get_osm(way(3810479), source = src, full = TRUE)

osmar object
11 nodes, 1 ways, 0 relations
```

The first statement retrieves the way only (because the default value of the `full` argument is `FALSE`). The second statement additionally retrieves all nodes that are members of the way (i.e., all nodes that define the way).

The second possibility to retrieve elements is to specify a bounding box by defining the left, bottom, right, and top coordinates (`corner_bbox()`), or the center point and width and height in meters (`center_bbox()`):

```
> bb <- center_bbox(174.76778, -36.85056, 700, 700)
> ua <- get_osm(bb, source = src)
> ua

osmar object
2427 nodes, 428 ways, 70 relations
```

The use of planet files via Osmosis as source works analogously. The source is specified by the function `osmsource_osmosis()`. The function's two arguments are the path to the planet file (`file`) and the path to the 'osmosis' tool (`osmosis = "osmosis"`). Note that per default it is assumed that the Osmosis executable is in your 'PATH' environment variable. The navigator example demonstrates the usage of planet files.

Working with the data

The retrieved `osmar` object is a list with the three elements `nodes`, `ways`, and `relations`. Each element again is a list containing `data.frames` for the attributes (the `attrs` list element) and meta-data (the `tags` list element) of the OSM elements. Ways and relations additionally have a `data.frame` containing their members (the `refs` list element).

Summarize. For each element `nodes`, `ways`, and `relations` of an `osmar` object an individual summary method is available. The overall summary method for an `osmar` object shows the three individual summaries all at once.

```
> summary(ua$nodes)

osmar$nodes object
2427 nodes, 771 tags

..$attrs data.frame:
    id, lat, lon, user, uid, visible, version,
    changeset, timestamp
..$tags data.frame:
    id, k, v

Bounding box:
          lat      lon
min -36.85661 174.7627
max -36.84472 174.7753

Key-Value contingency table:
             Key                   Value Freq
1      addr:city                Auckland  101
2    addr:street            Queen Street   61
3   addr:country                      NZ   40
4  addr:postcode                    1010   39
5        comment Determined via Keypa...   29
6    addr:street          Symonds Street   27
7        highway         traffic_signals   23
8    addr:street            Lorne Street   19
9        highway                bus_stop   15
10       amenity                    cafe   11
```

In the case of the summary for `nodes`, the number of elements and tags, as well as the available variables for each corresponding `data.frame` are shown. The bounding box of the coordinates and a contingency table of the top ten most frequently available key-value pairs are printed.

The summaries for the other two elements `ways` and `relations` are similar. Note that these methods in fact return the contingency table of all available key-value pairs and, in addition, further information which is not printed but may be useful for a descriptive analysis. We refer to the help pages (e.g., `?summary.nodes`) for a detailed description of the return values.

Find. In order to find specific elements within the `osmar` object, the `find()` function allows the object to be queried with a given condition. As the basis of `osmar` objects are `data.frames`, the condition

principally is a logical expression indicating the rows to keep. In addition, one has to specify to which element (nodes, `node()`; ways, `way()`; or relations, `relation()`) and to which data (attributes, `attrs()`; meta-data, `tags()`; or members, `refs()`) the condition applies.

If, for example, we want to find all traffic signal nodes, we know from the object's summary that the corresponding value in the `attrs data.frame` is `"traffic_signals"`. We can express this condition as follows:

```
> ts_ids <- find(ua, node(tags(v == "traffic_signals")))
> ts_ids

[1] 25769635 25769637 25769641 ...
```

The result is a vector with node IDs (in this case 25 traffic signal nodes) or `NA`. If the correct spelling is unknown, the defined binary operators `%agrep%` for approximate matches (see `?agrep`) and `%grep%` for pattern matches (see `?grepl`) can be used:

```
> bs_ids <- find(ua, node(tags(v %agrep% "busstop")))
> bs_ids

[1] 678301119  737159091  1318401034 ...
```

This returns 15 bus stops available in the `ua osmar` object.

We use the functions `find_down()` and `find_up()` to find all related elements for given element IDs. The OSM basic elements define a hierarchy,

$$\text{node} \leftarrow \text{way} \leftarrow \text{relation},$$

and these two functions enable us to find the related elements up and down the hierarchy. For example, `find_up()` on a node returns all related nodes, ways, and relations; `find_down()` on a node returns only the node itself. On the other hand, `find_up()` on a relation returns only the relation itself; `find_down()` on a relation returns the relation and all related ways and nodes.

```
> hw_ids <- find(ua, way(tags(k == "highway")))
> hw_ids <- find_down(ua, way(hw_ids))
```

In this example we find all ways that have a tag with the `k` attribute set to `"highway"`. These contain hardened and recognised land routes between two places used by motorised vehicles, pedestrians, cyclists, etc. The return value of `find_down()` and `find_up()` is a list containing the element IDs:

```
> str(hw_ids)

List of 3
 $ node_ids    : num [1:1321] 25769641 ...
 $ way_ids     : num [1:253] 4309608 ...
 $ relation_ids: NULL
```

Subset. The return value of the find functions then can be used to create subsets of `osmar` objects. The `subset()` method for `osmar` objects takes element IDs and returns the corresponding data as `osmar` objects. For example, the two subsets based on the traffic signal and bus stop element IDs are:

```
> ts <- subset(ua, node_ids = ts_ids)
> ts

osmar object
25 nodes, 0 ways, 0 relations

> bs <- subset(ua, node_ids = bs_ids)
> bs

osmar object
15 nodes, 0 ways, 0 relations
```

The subset based on the highway element IDs is:

```
> hw <- subset(ua, ids = hw_ids)
> hw

osmar object
1321 nodes, 253 ways, 0 relations
```

Note that the subsetting of osmar objects is divided into the two steps "finding" and "subsetting" to have more flexibility in handling the related elements (here with using `find_down()` and `find_up()`, but more sophisticated routines can be imagined).

Plot. The visualization of osmar objects is possible if nodes are available in the object (as only these OSM elements contain latitude and longitude information). The functions `plot_nodes()` and `plot_ways()` plot the available nodes as dots and ways as lines, respectively. The `plot()` method combines these two function calls. Note that this is a plot of the raw data and no projection is done (see the following section for a projected visualization).

```
> plot(ua)
> plot_ways(hw, add = TRUE, col = "green")
> plot_nodes(ts, add = TRUE, col = "red")
> plot_nodes(bs, add = TRUE, col = "blue")
```

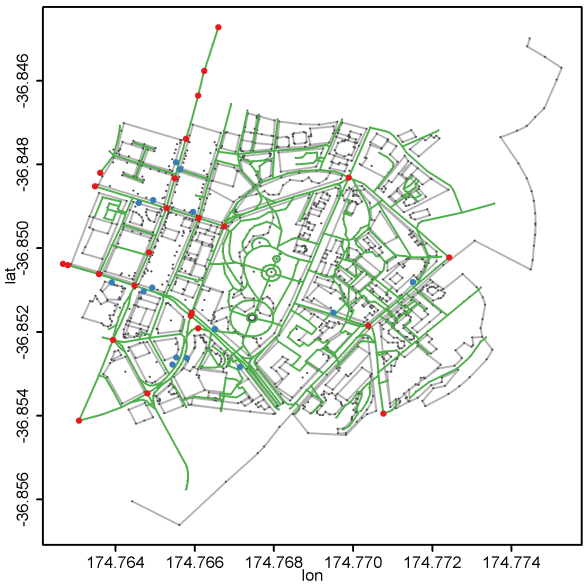


Figure 2: University of Auckland; roads are green lines; bus stops are blue and traffic signals are red points.

Converting the data

In order to use the complete power of R on OpenStreetMap data, it is essential to be able to convert osmar objects into commonly used objects based on classes provided by other packages. Currently, **osmar** provides two converters – into the **sp** (Bivand et al., 2008) and the **igraph** (Csardi, 2011) packages. In this section we show the conversion to sp objects, the navigation device example shows the conversion to igraph objects.

The **sp** package provides special data structures and utility functions for spatial data. Spatial data classes are available for points, lines, and polygons and others (see Bivand et al., 2008). The **osmar** package provides the `as_sp()` function,

```
> args(as_sp)

function(obj, what = c("points", "lines", "polygons"),
         crs = osm_crs(), simplify = TRUE)
NULL
```

to convert an osmar object into the corresponding classes for points, lines, and polygons in the **sp** package (given the required data are available). Note that the appropriate WGS84 coordinate reference system (CRS) for OpenStreetMap data is used (cf. `osm_crs()`).

Polygons. Polygons are used to represent areas, which OpenStreetMap represents as closed ways. Buildings, for example, are closed ways and can be converted to an sp polygon object:

```
> bg_ids <- find(ua, way(tags(k == "building")))
> bg_ids <- find_down(ua, way(bg_ids))
> bg <- subset(ua, ids = bg_ids)
> bg

osmar object
991 nodes, 110 ways, 0 relations

> bg_poly <- as_sp(bg, "polygons")
```

The result is a `SpatialPolygonsDataFrame` with the `osmar` object's attributes (the `attrs` element) as its data. Functionality provided by the **sp** package can now be used to analyze the OSM data; e.g., the `summary()` method or the `spplot()` method—the latter one, for example, to see how often each building was modified:

```
> spplot(bg_poly, c("version"))
```

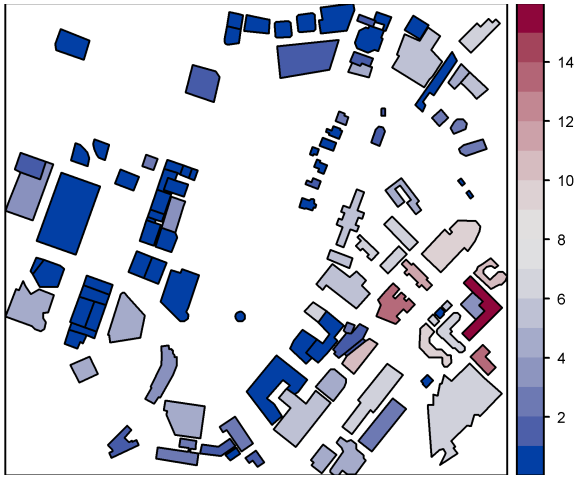


Figure 3: Number of modifications per building.

Meta-data (the `tags` element) and members (the `refs` element) are not automatically converted to a `SpatialPolygonsDataFrame`'s data.

Lines and points. The conversion of lines and points works similarly:

```
> hw_line <- as_sp(hw, "lines")
> bs_points <- as_sp(bs, "points")
```

The results are `SpatialLinesDataFrame` and `SpatialPointsDataFrame` objects, respectively.

In order to finalize the University of Auckland example we create a bus route map and visualize the available bus routes belonging to the bus stops. Therefore, we find all bus relations available in the object, retrieve the corresponding data from the OSM API, and convert the data into lines (note that this computation takes some time):

```
> bus_ids <- find(ua, relation(tags(v == "bus")))
> bus <- lapply(bus_ids,
+   function(i) {
+     raw <- get_osm(relation(i), full = TRUE)
+     as_sp(raw, "lines")
+   })
```

We use the argument `full = TRUE` to retrieve the relation itself and all related members. In detail, this means we retrieve all nodes, ways, and relations that are members of the specified relation; and, recursively, all nodes that are members of the retrieved ways.

We then use the **sp** plot methods to create the final bus route map:

```
> plot(bg_poly, col = "gray")
> plot(hw_line, add = TRUE, col = "green")
```

```
> plot(bs_points, add = TRUE, col = "blue")
> for ( i in seq(along = bus) ) {
+   plot(bus[[i]], add = TRUE, col = "blue")
+ }
```

Figure 4: Bus route map of the University of Auckland; roads are green lines; bus stops and bus routes are blue points and lines

.

R as navigator

We always wanted to know how a navigation device works. Now with **osmar**, R provides the necessary components and this serves as nice example on how to use **osmar**. The general idea is to (1) get the data, (2) extract all highways, (3) create a graph of all highway nodes with the distance between the highway nodes as edge weights, (4) compute the shortest path on the graph, and (5) trace the path on the highways.

Get the data. We use a planet file from Munich as the data source and use Osmosis (Henderson, 2011) to process the data. Note that 'osmosis' has to be in your 'PATH' environment variable.

```
> library("osmar")
> url <- "http://osmar.r-forge.r-project.org/"
> file <- "muenchen.osm.gz"
> download.file(sprintf("%s%s", url, file), file)
> system("gzip -d muenchen.osm.gz")
```

Get the center of Munich with a 3km × 3km bounding box:

```
> src <- osmsource_osmosis(file = "muenchen.osm")
> muc_bbox <- center_bbox(11.575278, 48.137222, 3000, 3000)
> muc <- get_osm(muc_bbox, src)
> muc

osmar object
13713 nodes, 3156 ways, 76 relations
```

For the navigation device we only need streets. This means, we have to find all ways that are tagged as highways and have a name tag, then find the associated nodes, and finally subset the full osmar object:

```
> hways_muc <- subset(muc, way_ids = find(muc, way(tags(k == "highway"))))
> hways <- find(hways_muc, way(tags(k == "name")))
> hways <- find_down(muc, way(hways))
> hways_muc <- subset(muc, ids = hways)
> hways_muc
```

```
osmar object
3889 nodes, 995 ways, 0 relations
```

Suppose we want to start our route at the "Sendlinger Tor". This means we first have to find a node that is tagged with the name "Sendlinger Tor" and then the nearest highway node:

```
> hway_start_node <- local({
+   id <- find(muc, node(tags(v == "Sendlinger Tor")))[1]
+   find_nearest_node(muc, id, way(tags(k == "highway")))
+ })
> hway_start <- subset(muc, node(hway_start_node))
```

For a given node (by its ID), the function `find_nearest_node()` finds the nearest node with the specified conditions (the package **geosphere**, Hijmans et al., 2011 is used to compute the distances). The end of the route should be in the northeast part of Munich; so we find nodes that are in the northeast and take one highway node:

```
> hway_end_node <- local({
+   id <- find(muc, node(attrs(lon > 11.59 & lat > 48.150)))[1]
+   find_nearest_node(muc, id, way(tags(k == "highway")))
+ })
> hway_end <- subset(muc, node(hway_end_node))
```

Finally, we visualize the initial situation:

```
> plot_nodes(muc, col = "gray")
> plot_ways(hways_muc, add = TRUE)
> plot_nodes(hways_muc, add = TRUE, col = "black")
> plot_nodes(hway_start, add = TRUE, col = "red")
> plot_nodes(hway_end, add = TRUE, col = "blue")
```

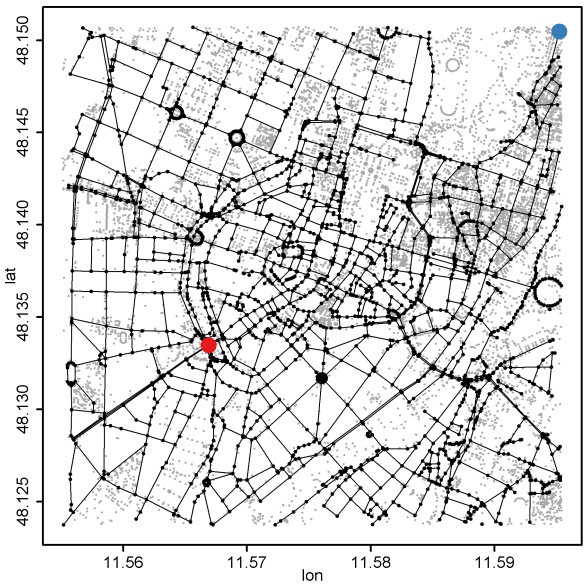


Figure 5: Highway map of Munich center.

The gray nodes are all nodes available in the full `osmar` object, the black nodes and lines are the road networks available in the highway-`osmar` object, the red and blue dots denote the starting and ending nodes of the searched route.

Compute the route. In order to compute the shortest route between the defined starting and ending nodes, we convert the highway-`osmar` object into a graph. R provides a set of packages to work with graphs, we decided to use **igraph**:

```
> library("igraph0")
> gr_muc <- as_igraph(hways_muc)
> summary(gr_muc)
```

```
Vertices: 2381
Edges: 2888
Directed: TRUE
No graph attributes.
Vertex attributes: name.
Edge attributes: weight, name.
```

The osmar object nodes define the nodes of the graph (node IDs are used as graph node names). The osmar object ways define the edges (way IDs are used as edge names), and the weights of the edges are the geographical distance between the nodes.

The **igraph** package provides different shortest path algorithms (e.g., Dijkstra and Bellman-Ford) via the function `get.shortest.paths()`. The shortest route (not necessarily unique) is:

```
> route <- get.shortest.paths(gr_muc,
+   from = as.character(hway_start_node),
+   to = as.character(hway_end_node))[[1]]
> route_nodes <- as.numeric(V(gr_muc)[route]$name)
```

We construct a new osmar object containing only elements related to the nodes defining the route:

```
> route_ids <- find_up(hways_muc, node(route_nodes))
> route_muc <- subset(hways_muc, ids = route_ids)
> route_muc

osmar object
101 nodes, 83 ways, 0 relations
```

And add the route to the highway map of Munich center in Figure 5:

```
> plot_nodes(route_muc, add = TRUE, col = "green")
> plot_ways(route_muc, add = TRUE, col = "green")
```

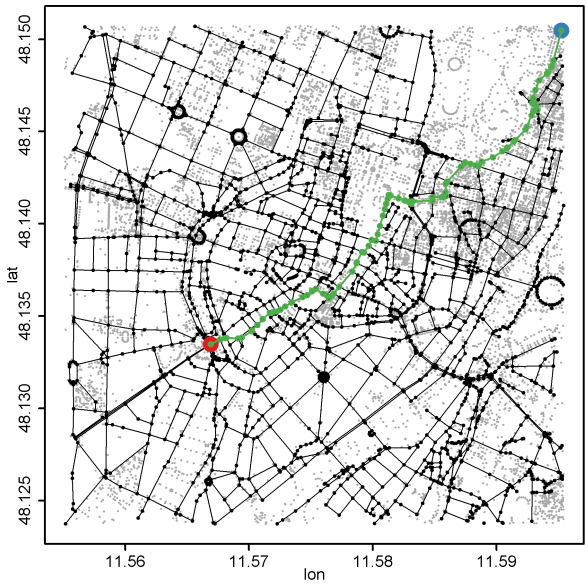


Figure 6: Shortest route on highway map of Munich center.

Route details. In order to present route details like street names, distances, and directions we have to work directly on the internals of the osmar objects.

We start by extracting the route's node IDs (which are in the correct order) and the way IDs (which we have to order) where the nodes are members:

```
> node_ids <- route_muc$nodes$attrs$id
> way_ids <- local({
+   w <- match(node_ids, route_muc$ways$refs$ref)
+   route_muc$ways$refs$id[w]
+ })
```

Then we extract the names of the ways in the correct order:

```
> way_names <- local({
+   n <- subset(route_muc$ways$tags, k == "name")
+   n[match(way_ids, n$id), "v"]
+ })
```

The next step is to extract the nodes' coordinates,

```
> node_coords <- route_muc$nodes$attrs[, c("lon", "lat")]
```

and to compute the distances (meters) and the bearings (degrees) between successive nodes (using the package **geosphere**):

```
> node_dirs <- local({
+   n <- nrow(node_coords)
+   from <- 1:(n-1)
+   to <- 2:n
+
+   cbind(dist = c(0,
+         distHaversine(node_coords[from, ], node_coords[to, ])),
+         bear = c(0,
+         bearing(node_coords[from, ],
+                 node_coords[to, ])))
+ })
```

Finally, we pack together all the information, and additionally compute the cumulative distance and a 16-point compass rose direction (the `compass()` function is available in the "navigator" demo from the **osmar** package):

```
> route_details <- data.frame(way_names, node_dirs)
> route_details$cdist <- cumsum(route_details$dist)
> route_details$dir <- compass(route_details$bear)
```

The result is a `data.frame` with a row for each node of the route. The row shows the name of the associated way, the distance (meters) and bearing (degrees and compass rose) to the successive node of the route, and the cumulative distance:

```
> head(route_details)
```

	way_names	dist	bear	cdist	dir
1	Sendlinger-Tor-Platz	0	0	0	N
2	Wallstraße	65	62	65	ENE
3	Herzog-Wilhelm-Straße	29	75	94	ENE
4	Oberanger	10	78	104	ENE
5	Oberanger	69	94	173	E
6	Nikolaus-Gradl-Weg	25	76	198	ENE

Needless to say that this navigation device can be made much more sophisticated. The **osmar** package contains the complete source code of the basic navigation device as a demo and we invite everybody to improve R as a navigator.

Summary

The **osmar** package extends the R ecosystem with infrastructure to work together with the OpenStreetMap project. So far, functionality is available to get data from different sources (e.g., planet file and API v0.6), to consolidate the data as an R `osmar` object, to work with the `osmar` object (e.g., subsetting and plotting), and to convert it to objects based on classes provided by other packages (e.g., to `igraph` and `sp` objects).

Future work includes the implementation of further converters; e.g., a converter from `osmar` objects to raster image objects via the **OpenStreetMap** package. We are also interested in implementing converters from objects provided by other R packages to `osmar` objects and in saving these objects in different OpenStreetMap sources via a `put_osm()` function. This would be, in fact, the completion of the **osmar** concept illustrated in Figure 1 with arrows from the right to the left blocks.

Furthermore, we would like to incorporate tools originated in the OpenStreetMap ecosystem. One idea is the implementation of a rule-based rendering tool for generating SVG images of OSM data along the lines of Osmarender (Topf, 2011). Another interesting project is Osmium, a fast and flexible

C++ and Javascript toolkit and framework for working with OSM data (Topf, 2012). An R interface (potentially via **Rcpp** modules; Eddelbuettel and François, 2011) would provide a very fast and flexible way to work with large OSM data sets.

Acknowledgment

The authors thank two anonymous reviewers and Joe Sakshaug for their constructive comments to improve the manuscript.

Bibliography

R. S. Bivand, E. J. Pebesma, and V. Gomez-Rubio. *Applied Spatial Data Analysis with R*. Springer, NY, 2008. URL http://www.asdar-book.org/. [p57]

G. Csardi. **igraph***: Network Analysis and Visualization*, 2011. URL http://cran.r-project.org/package=igraph. R package version 0.5.5-2. [p57]

D. Eddelbuettel and R. François. **Rcpp**: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL http://www.jstatsoft.org/v40/i08/. [p63]

I. Fellows. **OpenStreetMap***: Access to OpenStreetMap Raster Images*, 2012. URL http://CRAN.R-project.org/package=OpenStreetMap. R package version 0.2. [p53]

B. Henderson. *osmosis*, 2011. URL http://wiki.openstreetmap.org/wiki/Osmosis. Java application version 0.39. [p53, 59]

R. J. Hijmans, E. Williams, and C. Vennes. **geosphere***: Spherical Trigonometry*, 2011. URL http://CRAN.R-project.org/package=geosphere. R package version 1.2-24. [p60]

D. Kahle and H. Wickham. **ggmap***: A Package for Spatial Visualization with Google Maps and OpenStreetMap*, 2012. URL http://CRAN.R-project.org/package=ggmap. R package version 2.1. [p53]

M. Loecher. **RgoogleMaps***: Overlays on Google Map Tiles in R*, 2012. URL http://CRAN.R-project.org/package=RgoogleMaps. R package version 1.2.0. [p53]

OSM Foundation. *The OpenStreetMap Project*, 2011. URL http://openstreetmap.org. [p53]

T. Schlesinger and M. J. A. Eugster. **osmar***: OpenStreetMap and R*, 2012. URL http://cran.r-project.org/package=osmar. R package version 1.1. [p53]

J. Topf. *osmarender*, 2011. URL http://wiki.openstreetmap.org/wiki/Osmarender. XSLT transformation scripts. [p62]

J. Topf. *Osmium*, 2012. URL http://wiki.openstreetmap.org/wiki/Osmium. Fast and flexible C++ and Javascript toolkit and framework for working with OSM data. [p63]

Manuel J. A. Eugster
Institut für Statistik
Ludwig-Maximilians-Universität München
Ludwigstrasse 33, 80539 Munich
Germany
manuel.eugster@stat.uni-muenchen.de

Thomas Schlesinger
Institut für Statistik
Ludwig-Maximilians-Universität München
Ludwigstrasse 33, 80539 Munich
Germany
tho.schlesinger@googlemail.com

ftsa: An R Package for Analyzing Functional Time Series

by Han Lin Shang

Abstract Recent advances in computer recording and storing technology have tremendously increased the presence of functional data, whose graphical representation can be infinite-dimensional curve, image, or shape. When the same functional object is observed over a period of time, such data are known as functional time series. This article makes first attempt to describe several techniques (centered around functional principal component analysis) for modeling and forecasting functional time series from a computational aspect, using a readily-available R addon package. These methods are demonstrated using age-specific Australian fertility rate data from 1921 to 2006, and monthly sea surface temperature data from January 1950 to December 2011.

Introduction

The aim of this article is to describe the R functions that are readily-available in the **ftsa** package (Hyndman and Shang, 2013), for modeling and forecasting functional time series. This article was motivated by recent advances in computer recording and storing technology that have enabled researchers to collect and store (ultra) high-dimensional data. When the high-dimensional data are repeatedly measured on the same object over a period of time, a time series of continuous functions is observed within a common bounded interval (Shen and Huang, 2008).

Analyzing functional time series has received increasing attention in the functional data analysis literature (see for example, Hörmann and Kokoszka, 2010; Horváth et al., 2010; Horváth and Kokoszka, 2012). Hyndman and Shang (2010) presented a rainbow plot for visualizing functional time series, where the distant past data are shown in red and most recent data are shown in purple. Aguilera et al. (1999) proposed functional principal component regression (FPCR) to model and forecast functional time series.

Before reviewing the FPCR, we first define the problem more precisely. Let $y_t(x)$ denote a function, such as age-specific fertility rates for the continuous age variable x in year t, or monthly sea surface temperatures for the continuous time variable x in year t. In the latter example, functional time series techniques allow us to capture the underlying dynamic of the multiple seasonality in the data (see Shang and Hyndman, 2011; Shang, 2013, for example). We assume that there is an underlying smooth function $f_t(x)$ that observes with error at discretized grid points of x. In practice, we observe $\{x_i, y_t(x_i)\}$ for $t = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, p$, from which we extract a smooth function $f_t(x)$, given by

$$y_t(x_i) = f_t(x_i) + \sigma_t(x_i)\varepsilon_{t,i}, \tag{1}$$

where $\varepsilon_{t,i}$ is an independent and identically distributed (iid) standard normal random variable, $\sigma_t(x_i)$ allows the amount of noise to vary with x_i, and $\{x_1, x_2, \ldots, x_p\}$ is a set of discrete data points. Given a set of functional data $\boldsymbol{f}(x) = [f_1(x), f_2(x), \ldots, f_n(x)]^\top$, we are interested in finding underlying patterns using the FPCR, from which we obtain forecasts of $y_{n+h}(x)$, where h denotes the forecast horizon.

This article proceeds as follows. Techniques for modeling and forecasting functional time series are reviewed and their implementations using the **ftsa** package are described. Conclusions are given at the end.

Functional time series modeling and forecasting techniques

Functional principal component regression

The theoretical, methodological and practical aspects of functional principal component analysis (FPCA) have been extensively studied in the functional data analysis literature, since it allows finite dimensional analysis of a problem that is intrinsically infinite-dimensional (Hall and Hosseini-Nasab, 2006). Numerous examples of using FPCA as an estimation tool in regression problem can be found in different fields of applications, such as breast cancer mortality rate modeling and forecasting (Erbas et al., 2007), call volume forecasting (Shen and Huang, 2008), climate forecasting (Shang and Hyndman, 2011), demographical modeling and forecasting (Hyndman and Shang, 2009), and electricity demand forecasting (Antoch et al., 2008).

At a population level, a stochastic process denoted by f can be decomposed into the mean function and the sum of the products of orthogonal functional principal components and uncorrelated principal component scores. It can be expressed as

$$f = \mu + \sum_{k=1}^{\infty} \beta_k \phi_k,$$

where μ is the unobservable population mean function, β_k is the k^{th} principal component scores, and ϕ_k is the k^{th} population functional principal component.

In practice, we can only observe n realizations of f evaluated on a compact interval $x \in [0, \tau]$, denoted by $f_t(x)$ for $t = 1, 2, \ldots, n$. At a sample level, the functional principal component decomposition can be written as

$$f_t(x) = \bar{f}(x) + \sum_{k=1}^{K} \widehat{\beta}_{t,k} \widehat{\phi}_k(x) + \widehat{\varepsilon}_t(x), \tag{2}$$

where $\bar{f}(x) = \frac{1}{n}\sum_{t=1}^{n} f_t(x)$ is the estimated mean function, $\widehat{\phi}_k(x)$ is the k^{th} estimated orthonormal eigenfunction of the empirical covariance operator

$$\widehat{\Gamma}(x) = \frac{1}{n}\sum_{t=1}^{n}[f_t(x) - \bar{f}(x)][f_t(x) - \bar{f}(x)],$$

the coefficient $\widehat{\beta}_{t,k}$ is the k^{th} principal component score for year t, it is given by the projection of $f_t(x) - \bar{f}(x)$ in the direction of k^{th} eigenfunction $\widehat{\phi}_k(x)$, that is, $\widehat{\beta}_{t,k} = < f_t(x) - \bar{f}(x), \widehat{\phi}_k(x) > = \int_x [f_t(x) - \bar{f}(x)]\widehat{\phi}_k(x)dx$, $\widehat{\varepsilon}_t(x)$ is the residual, and K is the optimal number of components, which can be chosen by cross validation. Hyndman and Booth (2008) studied the impact on forecast accuracy with a smaller or larger than the optimal value of K.

The functional principal component decomposition is first demonstrated using the age-specific Australian fertility rate data between ages 15 and 49 observed from 1921 to 2006. This data set was obtained from the Australian Bureau of Statistics (Cat No, 3105.0.65.001, Table 38). A functional graphical display is given in Shang (2011).

Figure 1 presents the first two functional principal components and their associated principal component scores. The bottom panel of Figure 1 also plots the forecasted principal component scores, and their 80% prediction intervals (in yellow color), using an exponential smoothing state-space model (Hyndman et al., 2008). As pointed out by a referee, the forecasts of principal component scores appear to quickly level off, and the prediction intervals widen very quickly. This reflects the difficulty of our model in forecasting medium or long term horizon, as a result of the increase in variability.

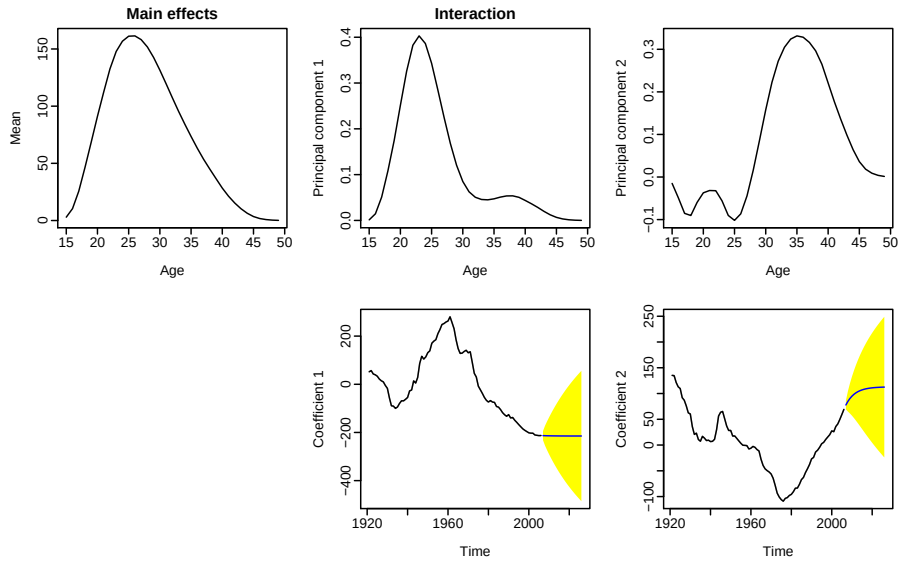


Figure 1: The first two functional principal components and their associated principal component scores for the Australian fertility rate data from 1921 to 2006.

Figure 1 was produced by the following code.

```
# load the package used throughout this article
library("ftsa")
```

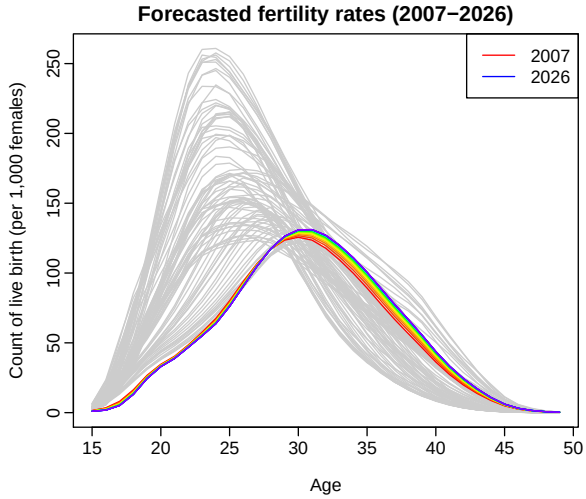
```
# Fit and plot functional principal components
# order specifies the number of principal components
# h specifies the forecast horizon
plot(forecast(ftsm(Australiasmoothfertility, order=2), h=20), "components")
```

By conditioning on the set of smoothed functions $\boldsymbol{f}(x) = [f_1(x), f_2(x), \ldots, f_n(x)]^\top$ and the fixed functional principal components $\mathcal{B} = [\widehat{\phi}_1(x), \widehat{\phi}_2(x), \ldots, \widehat{\phi}_K(x)]^\top$, the h-step-ahead forecasts of $y_{n+h}(x)$ can be obtained as

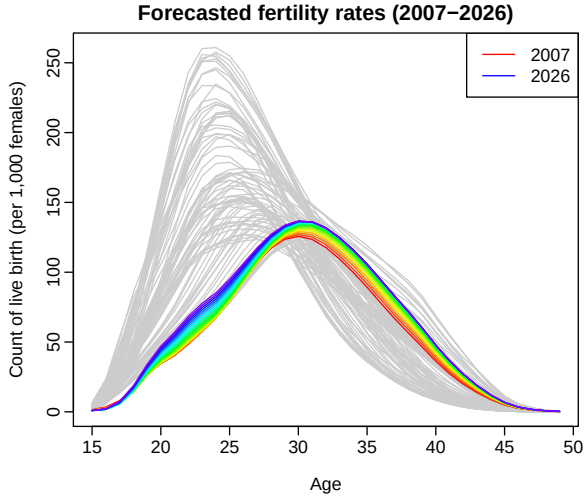
$$\widehat{y}_{n+h|n}(x) = \mathrm{E}[y_{n+h}(x)|\boldsymbol{f}(x), \mathcal{B}] = \bar{f}(x) + \sum_{k=1}^{K} \widehat{\beta}_{n+h|n,k}\widehat{\phi}_k(x),$$

where $\widehat{\beta}_{n+h|n,k}$ denotes the h-step-ahead forecasts of $\beta_{n+h,k}$ using a univariate time series.

Figure 2 shows the forecasts of Australian fertility rate data from 2007 to 2026 highlighted in rainbow color, while the data used for estimation are grayed out. Both the multi-step-ahead and iterative one-step-ahead forecasts exhibit a continuing shift to older ages of peak fertility rates, caused by a recent tendency to postpone child-bearing while pursuing careers.



(a) Multiple-step-ahead forecasts. Based on the historical data from 1921 to 2006, we obtain 20-step-ahead forecasts for 2007 to 2026.



(b) Iterative one-step-ahead forecasts. Based on the historical data from 1921 to 2006, we obtain iterative one-step-ahead forecasts for 2007 to 2026 using the rolling origin approach.

Figure 2: Forecasts of the Australian fertility rates from 2007 to 2026, based on the first two functional principal components and their associated principal component scores as an illustration.

Figure 2 was produced by the following code.

```
# Plot the historical data in gray
plot(Australiasmoothfertility, col = gray(0.8), xlab = "Age",
     ylab = "Count of live birth (per 1,000 females)",
     main = "Forecasted fertility rates (2007-2026)")
# Plot the forecasts in rainbow color for Fig. 4(a)
plot(forecast(ftsm(Australiasmoothfertility, order = 2), h = 20), add = TRUE)
legend("topright", c("2007", "2026"), col = c("red", "blue"), lty = 1)

plot(Australiasmoothfertility, col = gray(0.8), xlab = "Age",
     ylab = "Count of live birth (per 1,000 females)",
     main = "Forecasted fertility rates (2007-2026)")
# Plot the forecasts in rainbow color for Fig. 4(b)
plot(ftsmiterativeforecasts(Australiasmoothfertility, components = 2, iteration = 20),
     add = TRUE)
legend("topright", c("2007", "2026"), col = c("red", "blue"), lty = 1)
```

To construct prediction interval, we calculate the forecast variance that follows from (1) and (2). Because of orthogonality, the forecast variance can be approximated by the sum of component variances

$$\xi_{n+h}(x) = \text{Var}[y_{n+h}(x)|\boldsymbol{f}(x), \mathcal{B}] = \widehat{\sigma}_{\mu}^2(x) + \sum_{k=1}^{K} u_{n+h,k}\widehat{\phi}_k^2(x) + v(x) + \sigma_{n+h}^2(x),$$

where $u_{n+h,k} = \text{Var}(\beta_{n+h,k}|\beta_{1,k}, \beta_{2,k}, \ldots, \beta_{n,k})$ can be obtained from the time series model, and the model error variance $v(x)$ is estimated by averaging $\{\widehat{\varepsilon}_1^2(x), \widehat{\varepsilon}_2^2(x), \ldots, \widehat{\varepsilon}_n^2(x)\}$ for each x, and $\widehat{\sigma}_{\mu}^2(x)$ and $\sigma_{n+h}^2(x)$ can be obtained from the nonparametric smoothing method used.

Based on the normality assumption, the $100(1-\alpha)\%$ prediction interval for $y_{n+h}(x)$ is constructed as $\widehat{y}_{n+h|n}(x) \pm z_{\alpha}\sqrt{\xi_{n+h}(x)}$, where z_{α} is the $(1-\alpha/2)$ standard normal quantile.

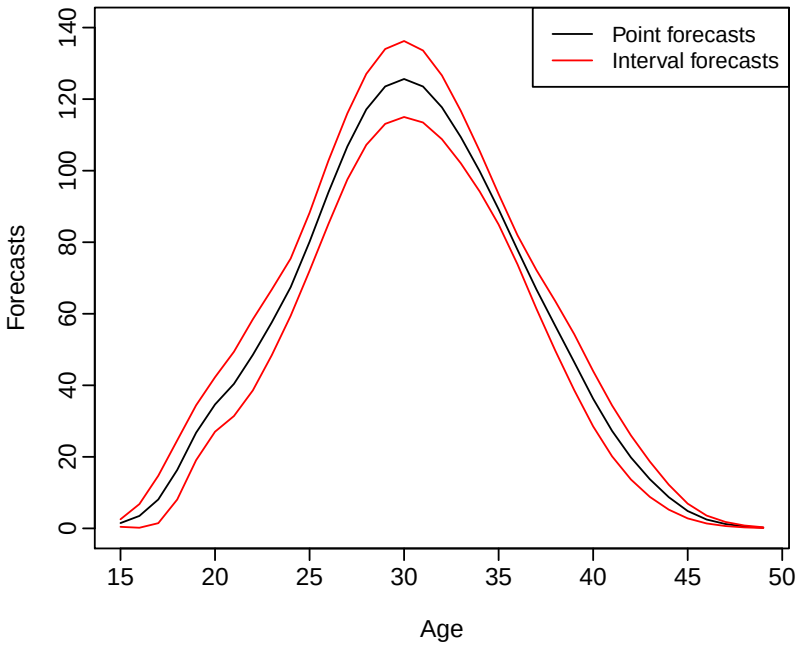


Figure 3: Forecasts of fertility rates in 2007, along with the 80% prediction interval.

Figure 3 displays the forecasts of fertility rates in 2007, along with the 80% prediction interval. It was created by the following code.

```
# Plot the point forecast
aus = forecast(ftsm(Australiasmoothfertility, order = 2), h = 1)
plot(aus, ylim = c(0, 140))
# Plot the lower and upper bounds
lines(aus$lower, col = 2); lines(aus$upper, col = 2)
# Add a legend to the plot
```

```
legend("topright", c("Point forecasts", "Interval forecasts"), col = c(1, 2), lty = 1,
       cex = 0.9)
```

For this Australian fertility rate data, the point and interval forecast accuracy obtained by the FPCR have already been studied by Shang (2012).

Updating point and interval forecasts

A special case of functional time series is when the continuous variable is also a time variable, such as the monthly sea surface temperature data from 1950 to 2011, obtained from National Oceanic and Atmospheric Administration (http://www.cpc.noaa.gov/data/indices/sstoi.indices). A similar type of functional graphical display is given in Shang (2011, Figure 2). Such data originate from a univariate seasonal time series. Let $\{Z_w, w \in [1, N]\}$ be a seasonal time series which has been observed at N equispaced times. We divide the observed time series into n trajectories, and then consider each trajectory of length p as a curve rather than p distinct data points. The functional time series is given by

$$y_t(x) = \{Z_w, w \in (p(t-1), pt]\}, \qquad t = 1, 2, \ldots, n.$$

The problem of interest is to forecast $y_{n+h}(x)$, where h denotes forecast horizon. In the sea surface temperature data, we consider $\{Z_w\}$ to be monthly sea surface temperatures from 1950 to 2011, so that $p = 12$ and $N = 62 \times 12 = 744$, and we are interested in forecasting sea surface temperatures in 2012 and beyond.

When $N = np$, all trajectories are complete, and forecasts can be obtained by the FPCR. However, when $N \neq np$, we revisited the block moving (BM) and penalized least squares (PLS) proposed by Shang and Hyndman (2011) to update point and interval forecasts, when the most recent curve is partially observed.

When functional time series are segments of a univariate time series, the most recent trajectory is observed sequentially (Hyndman and Shang, 2010). When we observe the first m_0 time period of $y_{n+1}(x_l)$, denoted by $\boldsymbol{y}_{n+1}(x_e) = [y_{n+1}(x_1), y_{n+1}(x_2), \ldots, y_{n+1}(x_{m_0})]^\top$, we are interested in forecasting the data in the remaining time period, denoted by $y_{n+1}(x_l)$ for $m_0 < l \leq p$. By using the FPCR, the partially observed data in the most recent curve are not incorporated into the forecasts of $y_{n+1}(x_l)$. Indeed, the point forecasts obtained from the FPCR can be expressed as

$$\widehat{y}_{n+1|n}(x_l) = \mathrm{E}[y_{n+1}(x_l)|\boldsymbol{f}(x_l), \mathcal{B}_l] = \bar{f}(x_l) + \sum_{k=1}^{K} \widehat{\beta}_{n+1|n,k}\widehat{\phi}_k(x_l),$$

for $m_0 < l \leq p$, where $\boldsymbol{f}(x_l)$ denotes the historical data corresponding to the remaining time periods; $\bar{f}(x_l)$ is the mean function corresponding to the remaining time periods; and $\mathcal{B}_l = \{\widehat{\phi}_1(x_l), \widehat{\phi}_2(x_l), \ldots, \widehat{\phi}_K(x_l)\}$ is a set of the estimated functional principal components corresponding to the remaining time periods.

In order to improve point forecast accuracy, it is desirable to dynamically update the point and interval forecasts for the rest of year $n + 1$ by incorporating the partially observed data. In what follows, I shall revisit two methods for updating point and interval forecasts.

Block moving (BM)

The BM method re-defines the start and end points of trajectories. Because time is a continuous variable, we can change the function support from $[1, p]$ to $[m_0 + 1, p] \bigcup [1, m_0]$. The re-defined functional time series forms a complete block, at the cost of losing some observations in the first year. With the complete data block, the FPCR can then be applied to update the point and interval forecasts.

The re-defined data are shown diagrammatically in Figure 4, where the bottom box has moved to become the top box. The cyan colored region shows the data loss in the first year. The partially observed last trajectory under the old "year" completes the last trajectory under the new year.

As an illustration, suppose we observe the monthly sea surface temperature data from January 1950 to May 2011, we aim to update the point and interval forecasts from June 2011 to December 2011. Figure 5 displays the point and interval forecasts for the remaining months of 2011, by using the BM method.

Figure 5 was created by the following code

```
# Name history to represent historical data,
history <- ElNino2011smooth
# Name obs to represent partially observed data,
obs <- ElNino2011smooth$y[1:5,62]
```

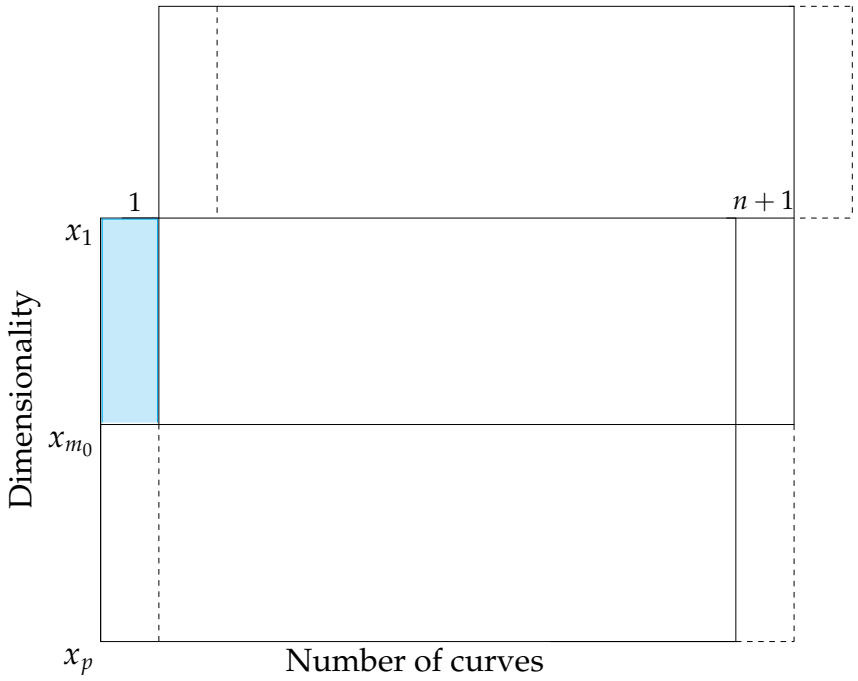


Figure 4: Dynamic update via the BM approach. The colored region shows the data loss in the first year. The forecasts for the remaining months in year $n+1$ can be updated by the forecasts using the TS method applied to the upper block.

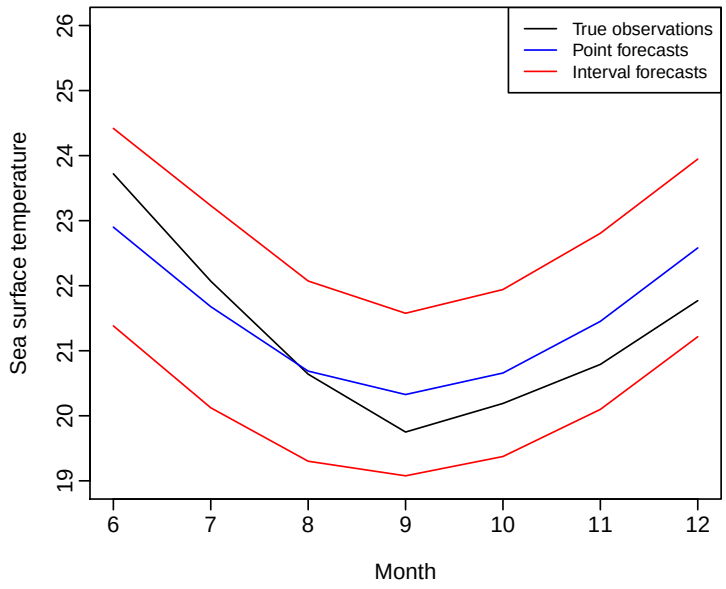


Figure 5: Prediction interval of the sea surface temperature data between June 2011 and December 2011. By incorporating the sea surface temperature data between January 2011 and May 2011, the 80% prediction interval can be updated using the BM method.

```
# Name fore to represent the forecasting period
fore <- ElNino2011smooth$y[6:12,62]
int <- dynupdate(data = history, newdata = obs, holdoutdata = fore,
  method = "block", interval = TRUE, level = 80)
bmupdate <- dynupdate(data = history, newdata = obs, holdoutdata = fore,
  method = "block", value = TRUE)
plot(6:12, fore, type = "l", ylim = c(19, 26), xlab = "Month",
  ylab = "Sea surface temperature")
```

```
lines(6:12, bmupdate, col = 4)
lines(6:12, int$low$y, col = 2); lines(6:12, int$up$y, col = 2)
legend("topright", c("True observations", "Point forecasts", "Interval forecasts"),
   col=c(1, 4, 2), lty = 1, cex = 0.8)
```

Penalized least squares (PLS)

We can also update the remaining part of the trajectory by using regression-based approaches. Let $\boldsymbol{F}_e$ be $m_0 \times K$ matrix, whose $(j,k)^{\text{th}}$ entry is $\widehat{\phi}_k(x_j)$ for $1 \leq j \leq m_0$. Let $\boldsymbol{\beta}_{n+1} = [\beta_{n+1,1}, \beta_{n+1,2}, \ldots, \beta_{n+1,K}]^\top$, $\bar{\boldsymbol{f}}(x_e) = [\bar{f}(x_1), \bar{f}(x_2), \ldots, \bar{f}(x_{m_0})]^\top$, and $\boldsymbol{\epsilon}_{n+1}(x_e) = [\epsilon_{n+1}(x_1), \epsilon_{n+1}(x_2), \ldots, \epsilon_{n+1}(x_{m_0})]^\top$. As the mean-adjusted $\widehat{\boldsymbol{y}}^*_{n+1}(x_e) = \boldsymbol{y}_{n+1}(x_e) - \bar{\boldsymbol{f}}(x_e)$ become available, a regression can be expressed as

$$\widehat{\boldsymbol{y}}^*_{n+1}(x_e) = \boldsymbol{F}_e \boldsymbol{\beta}_{n+1} + \boldsymbol{\epsilon}_{n+1}(x_e).$$

The $\boldsymbol{\beta}_{n+1}$ can be estimated by ordinary least squares, assuming $(\boldsymbol{F}_e^\top \boldsymbol{F}_e)$ is invertible,

$$\widehat{\boldsymbol{\beta}}^{\text{OLS}}_{n+1} = (\boldsymbol{F}_e^\top \boldsymbol{F}_e)^{-1} \boldsymbol{F}_e^\top \widehat{\boldsymbol{y}}^*_{n+1}(x_e).$$

However, if $(\boldsymbol{F}_e^\top \boldsymbol{F}_e)$ is not invertible, then a regularized approach can be implemented, such as the ridge regression (RR) and penalized least squares (PLS). The regression coefficients of the RR and PLS are

$$\begin{aligned} \widehat{\boldsymbol{\beta}}^{\text{RR}}_{n+1} &= (\boldsymbol{F}_e^\top \boldsymbol{F}_e + \lambda \boldsymbol{I}_K)^{-1} \boldsymbol{F}_e^\top \widehat{\boldsymbol{y}}_{n+1}(x_e), \\ \widehat{\boldsymbol{\beta}}^{\text{PLS}}_{n+1} &= (\boldsymbol{F}_e^\top \boldsymbol{F}_e + \lambda \boldsymbol{I}_K)^{-1} (\boldsymbol{F}_e^\top \widehat{\boldsymbol{y}}_{n+1}(x_e) + \lambda \widehat{\boldsymbol{\beta}}_{n+1|n}), \end{aligned} \tag{3}$$

where $\widehat{\boldsymbol{\beta}}^{\text{RR}}_{n+1} \to \boldsymbol{0}$ as $\lambda \to \infty$, and $\widehat{\boldsymbol{\beta}}^{\text{RR}}_{n+1} \to \widehat{\boldsymbol{\beta}}^{\text{OLS}}_{n+1}$ as $\lambda \to 0$. In contrast, the $\widehat{\boldsymbol{\beta}}^{\text{PLS}}_{n+1} \to \widehat{\boldsymbol{\beta}}_{n+1|n}$ as $\lambda \to \infty$, and $\widehat{\boldsymbol{\beta}}^{\text{PLS}}_{n+1} \to \widehat{\boldsymbol{\beta}}^{\text{OLS}}_{n+1}$ as $\lambda \to 0$.

The point forecasts of $y_{n+1}(x_l)$ obtained by the RR and PLS methods are given by

$$\begin{aligned} \widehat{y}^{\text{RR}}_{n+1}(x_l) &= \bar{f}(x_l) + \sum_{k=1}^{K} \widehat{\beta}^{\text{RR}}_{n+1,k} \widehat{\phi}_k(x_l), \\ \widehat{y}^{\text{PLS}}_{n+1}(x_l) &= \bar{f}(x_l) + \sum_{k=1}^{K} \widehat{\beta}^{\text{PLS}}_{n+1,k} \widehat{\phi}_k(x_l). \end{aligned}$$

Among these regression-based approaches, the PLS method can also update the interval forecasts. Let the one-step-ahead forecast errors of the principal component scores be given by

$$\widehat{\xi}_{j,k} = \widehat{\beta}_{n-j+1,k} - \widehat{\beta}_{n-j+1|n-j,k}, \quad \text{for} \quad j = 1, 2, \ldots, n-K.$$

$\{\widehat{\xi}_{1,k}, \widehat{\xi}_{2,k}, \ldots, \widehat{\xi}_{n-K,k}\}$ can then be sampled with replacement to give a bootstrap sample of $\beta_{n+1|n,k}$:

$$\widehat{\beta}^b_{n+1|n,k} = \widehat{\beta}_{n+1|n,k} + \widehat{\xi}^b_{*,k}, \quad \text{for} \quad b = 1, 2, \ldots, B,$$

where $\widehat{\xi}^b_{*,k}$ denotes the bootstrap samples, and B is the number of bootstrap replications. Based on (3), the bootstrapped $\widehat{\boldsymbol{\beta}}^b_{n+1|n}$ leads to the bootstrapped $\widehat{\boldsymbol{\beta}}^{b,\text{PLS}}_{n+1}$, we obtain B replications of

$$\widehat{y}^{b,\text{PLS}}_{n+1}(x_l) = \bar{f}(x_l) + \sum_{k=1}^{K} \widehat{\beta}^{b,\text{PLS}}_{n+1,k} \widehat{\phi}_k(x_l) + \widehat{\epsilon}^b_{n+1}(x_l),$$

where $\widehat{\epsilon}^b_{n+1}(x_l)$ is obtained by sampling with replacement from $\{\widehat{\epsilon}_1(x_l), \widehat{\epsilon}_2(x_l), \ldots, \widehat{\epsilon}_n(x_l)\}$. Hence, the $100(1-\alpha)\%$ prediction interval for the updated forecasts are defined as $\alpha/2$ and $(1-\alpha/2)$ quantiles of $\widehat{y}^{b,\text{PLS}}_{n+1}(x_l)$.

Figure 6 displays the point and interval forecasts for the remaining time period of year 2011, by using the PLS method. It was created by the following code

```
history <- ElNino2011smooth
obs <- ElNino2011smooth$y[1:5, 62]
fore <- ElNino2011smooth$y[6:12, 62]
# Implement the ridge and PLS regressions,
# The tuning parameter lambda=100 as an
# illustration
```

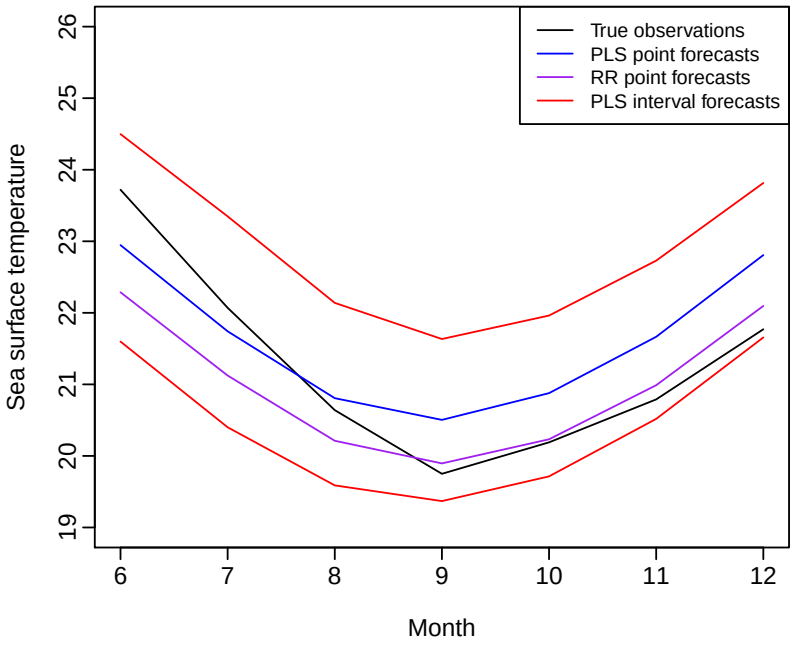


Figure 6: Prediction interval of the sea surface temperature data between June 2011 and December 2011. By incorporating the sea surface temperature data between January 2011 and May 2011, the 80% prediction interval can be updated using the PLS method.

```
rrmethod <- dynupdate(history, newdata = obs, holdoutdata = fore, method = "ridge",
  value = TRUE, lambda = 100, level = 80)
plsmethod <- dynupdate(history, newdata = obs, holdoutdata = fore, method = "pls",
  value = TRUE, lambda = 100, level = 80)
plsmethodint <- dynupdate(history, newdata = obs, holdoutdata = fore, method = "pls",
  interval = TRUE, lambda = 100, level = 80)
# Plot the true observations for forecasting period
plot(6:12, fore, type = "l", ylim = c(19, 26), xlab = "Month",
  ylab = "Sea surface temperature")
# Plot point forecasts obtained by RR and PLS
lines(6:12, plsmethod, col = 4); lines(6:12, rrmethod, col = "purple")
# Plot interval forecasts obtained by PLS
lines(6:12, plsmethodint$low$y, col = 2); lines(6:12, plsmethodint$up$y, col = 2)
legend("topright",c("True observations", "PLS point forecasts", "RR point forecasts",
  "PLS interval forecasts"), col = c(1, 4, "purple", 2), lty = 1, cex = 0.8)
```

For this sea surface temperature data set, the point and interval forecast accuracy obtained by the FPCR, BM and PLS methods have already been studied by Shang and Hyndman (2011).

Conclusion

This article described several techniques in the **ftsa** package, for modeling and forecasting functional time series. These methods centered on the FPCR, which is a common dimension reduction technique in the functional data analysis literature. FPCR reduces intrinsically infinite number of variables to several orthogonal regressors, which captures the main mode of variation in data. As illustrated by the Australian fertility rate data, FPCR is able to model and forecast annual Australian fertility rates through either multi-step-ahead forecasts or iterative one-step-ahead forecasts using the rolling origin approach. When the continuous variable in a functional time series is also a time variable, a new observation arrives sequentially. As shown using the monthly sea surface temperature data, the BM and PLS methods can update the point and interval forecasts based on the FPCR.

To sum up, the methods reviewed in this article focus on extracting patterns from a set of functional time series, and should be considered when the interest lies in modeling and forecasting the future realizations of a stochastic process.

Acknowledgements

I would like to thank the editors and two reviewers for constructive comments and suggestions that have significantly improved this article. Thanks also go to Professor Rob Hyndman for helping with R code.

Bibliography

A. Aguilera, F. Ocana, and M. Valderrama. Forecasting time series by functional PCA. Discussion of several weighted approaches. *Computational Statistics*, 14(3):443–467, 1999. [p64]

J. Antoch, L. Prchal, M. R. De Rosa, and P. Sarda. Functional linear regression with functional response: application to prediction of electricity consumption. In S. Dabo-Niang and F. Ferraty, editors, *Functional and Operatorial Statistics*, pages 23–29. Physica-Verlag, Heidelberg, 2008. [p64]

B. Erbas, R. J. Hyndman, and D. M. Gertig. Forecasting age-specific breast cancer mortality using functional data models. *Statistics in Medicine*, 26(2):458–470, 2007. [p64]

P. Hall and M. Hosseini-Nasab. On properties of functional principal components analysis. *Journal of the Royal Statistical Society: Series B*, 68(1):109–126, 2006. [p64]

S. Hörmann and P. Kokoszka. Weakly dependent functional data. *The Annals of Statistics*, 38(3): 1845–1884, 2010. [p64]

L. Horváth and P. Kokoszka. *Inference for Functional Data with Applications*. Springer, New York, 2012. [p64]

L. Horváth, M. Hušková, and P. Kokoszka. Testing the stability of the functional autoregressive process. *Journal of Multivariate Analysis*, 101(2):352–367, 2010. [p64]

R. Hyndman and H. L. Shang. *ftsa: Functional time series analysis*, 2013. URL http://CRAN.R-project.org/package=ftsa. R package version 3.8. [p64]

R. J. Hyndman and H. Booth. Stochastic population forecasts using functional data models for mortality, fertility and migration. *International Journal of Forecasting*, 24(3):323–342, 2008. [p65]

R. J. Hyndman and H. L. Shang. Forecasting functional time series (with discussion). *Journal of the Korean Statistical Society*, 38(3):199–221, 2009. [p64]

R. J. Hyndman and H. L. Shang. Rainbow plots, bagplots, and boxplots for functional data. *Journal of Computational and Graphical Statistics*, 19(1):29–45, 2010. [p64, 68]

R. J. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder. *Forecasting with Exponential Smoothing: the State Space Approach*. Springer, Berlin, 2008. [p65]

H. L. Shang. rainbow: an R package for visualizing functional time series. *The R Journal*, 3(2):54–59, 2011. [p65, 68]

H. L. Shang. Point and interval forecasts of age-specific fertility rates: a comparison of functional principal component methods. *Journal of Population Research*, 29(3):249–267, 2012. [p68]

H. L. Shang. Functional time series approach for forecasting very short-term electricity demand. *Journal of Applied Statistics*, 40(1):152–168, 2013. [p64]

H. L. Shang and R. J. Hyndman. Nonparametric time series forecasting with dynamic updating. *Mathematics and Computers in Simulation*, 81(7):1310–1324, 2011. [p64, 68, 71]

H. Shen and J. Z. Huang. Interday forecasting and intraday updating of call center arrivals. *Manufacturing & Service Operations Management*, 10(3):391–410, 2008. [p64]

Han Lin Shang
University of Southampton
Murray Building, University Road, Highfield
Southampton, SO17 1BJ, United Kingdom
H.Shang@soton.ac.uk

Statistical Software from a Blind Person's Perspective

R is the best, but we can make it better

by A. Jonathan R. Godfrey

Abstract Blind people have experienced access issues to many software applications since the advent of the Windows operating system; statistical software has proven to follow the rule and not be an exception. The ability to use R within minutes of download with next to no adaptation has opened doors for accessible production of statistical analyses for this author (himself blind) and blind students around the world. This article shows how little is required to make R the most accessible statistical software available today. There is any number of ramifications that this opportunity creates for blind students, especially in terms of their future research and employment prospects. There is potential for making R even better for blind users. The extensibility of R makes this possible through added functionality being made available in an add-on package called **BrailleR**. Functions in this package are intended to make graphical information available in text form.

A short introduction to general accessibility of computers for blind people

Access to computers for blind and vision impaired people is not easy to describe because the way each person does what they do is theirs alone. The level of visual acuity each blind person has may range from absolute zero useful vision through to an ability to read standard print, albeit using magnification of some description. The futility of making assumptions about how a blind person works is best demonstrated using a few examples. First, a person may be able to read 12pt font type on paper but have no access to the same size or larger print on a computer screen. Second, a person may be able to walk around their community in safety without use of a guide dog or white cane, but be entirely unable to read print in any form. For the purposes of this discussion, I use the term "blind" to indicate that a person has no ability to use print in any form, and "low vision" to mean that they can use print in some way; this second group is the harder to define and to cater for as their needs are much more diverse. Another reason for defining these terms is that the word "blind" does have a fairly consistent meaning across the world, but there is quite a lot of inconsistency about the terms used by those people who do not want to be labelled as blind, but do have some reduction in their vision compared to the majority of humans.

The vast majority of blind people use screen reading software. This software takes the text from keyboard entry or what is displayed on the screen and uses synthetic speech to read out as much as possible to the user. Screen reading software cannot handle graphics, except icons or images that are labelled with some substitute text; in web pages this text is known as an "alt tag".

In many circumstances, screen readers are used by people who have quite reasonable (but might still be described as low) vision as this option is faster than struggling with print based solutions. Use of a screen reader is however still considerably slower than use of normal vision. Users can become quite adept at increasing the speed of the speech to improve their efficiency, but reading speeds seldom surpass 500 words per minute. Limiting the amount of material that is read aloud by the software is therefore a key element in minimizing the amount of irrelevant information that needs to be processed by the user and would otherwise make them less efficient.

Depending on their level of vision, screen reader users can often also use braille displays or image enlargement technology to improve their experiences. Braille users can have reading speeds that approach the common reading speed of sighted people, but skimming unfamiliar text is difficult at best. The main limitation of braille is again that the access to graphics is basically nonexistent. Tactile image processing is being worked upon (Gardner et al., 2004; Watanabe et al., 2012), but universal access to the current technology is quite limited. Some examples of successful use of tactile images do exist (Ebert, 2005), but little success has occurred to date with the graphs found in statistical analyses. Attempts have also been made to convey the information presented in graphs using sounds (rather than speech); success in this endeavour has been limited to plots of mathematical functions rather than the not so well-behaved depictions of data presented in statistical graphs.

For those people who are able to use enlargement software or equipment, access to graphics is possible. In this respect and for the purposes of this presentation, these people are working as sighted people rather than as blind people and are therefore given little further consideration.

Whether it is through use of screen reader software or braille displays, most blind computer users will find new software at least as challenging to engage with as do their sighted peers. The familiarity of menu structures and the like gives as much comfort to the blind user as it does to the sighted user. The issue with new software comes from the use of a multitude of different style dialogue boxes and controls used by the software to achieve the desired outcome. The access to these dialogues is made even more difficult when non-standard controllers are used by software developers. Unfortunately, not all commonly available software development systems are designed to incorporate accessibility into software at the design phase. This means that the screen reader must try to interpret what is given on-screen, rather than being given clear instructions by the software. Icons are a simple example. When a new software application is designed and the creator chooses the standard icon for a common task, it is likely the screen reader already knows what that icon is for. Choosing an alternative that has some minor visual changes on the same theme as the standard icon, poses no challenge for the sighted user; for the blind user, this new icon will not be linked with the intended meaning until the screen reader is told how to interpret the new graphic.

Most blind people do not use the mouse to control their computer. Keyboard use, including key combinations to emulate mouse clicks, means that the blind user moves around menus and dialogue boxes using extra techniques they must learn. Poor design, mainly derived from inconsistent and therefore confusing structures in presentation, creates a problem for blind users coming to grips with new software. Another consideration is the amount of unused functionality built into many applications, which gives the unfamiliar user more work in manipulating dialogue boxes. The blind user who hopes to become less inefficient, quickly learns hot keys which reduce the time taken to perform the most basic tasks. Many well-designed applications show the user what these hot keys are, but many sighted computer users do not know that keyboard alternatives to mouse clicking do exist and that they could also benefit from their use.

Why is R the best?

Aside from the reasons a sighted R user might cite for preferring R over its alternatives, a blind user would add the following features that make R the superior statistical software option.

1. As suggested above, the blind user needs to minimize the amount of text that must be processed to maximise efficiency. The brevity of output that results from what I call a WYRIWYG (what you request is what you get) system is extremely useful for the blind user.
2. The R help pages are all available in HTML. It is not uncommon for help pages to be viewed using a web browser but screen reader software and web browsers already work well together. The help pages in R are simply constructed and consistent. Navigation is easy and the user can build familiarity that does not exist with help systems based on an archive of PDF documents. (More about PDF documents later.) It is also worthwhile mentioning the change in the way R provides a search of the help files since version 2.14.0, with the search results from the `??blah` command leading to a web page as against the former display which was not accessible.
3. R uses text based input and output. Some statistical software uses graphics windows to display simple text output. Screen readers are not sophisticated enough to extract text elements out of a graphics window. Often these objects are created using Java which on occasion can work with screen readers, but shared experiences of blind users are variable (to be kind).
4. R offers the user options for modes of interaction. The command line input using the keyboard (rather than mouse clicks and dialogue boxes) means R works with screen readers and braille displays that echo the typed input. It should be noted here that the dialogue boxes in the graphical user interface (GUI) form of R running under Windows do not all work well with screen readers. It is fortunate therefore that all dialogue box functionality is easily replicated using the command line.
5. R users commonly use script files that can be edited or created using any text editor. This might seem obvious, but while many statistical software applications have this feature, those that are primarily driven by mouse clicks and dialogue boxes are often used by people who cannot support the blind student needing to use commands entered via the keyboard. Documentation to support use of the command syntax for many statistical software options is not always kept up to date.
6. Access to R is there for anyone regardless of operating system, choice of screen reader, braille display, or financial resources. First, note that blindness is more prevalent in less developed countries where the price of R makes it a very attractive option. Second, in an education setting, sighted students can often use university-funded commercial software. A blind student who cannot use the software on university computers could incur extra costs to use statistical

software on their own computer which has the extra software and peripherals they need. In such settings, R provides a free back up plan for the blind student.

7. In situations where access is less than complete, such as PDF documents, the blind R user can fall back on the text documents that created the PDF. For example, *package vignettes* are additional documentation for packages provided as PDF documents with sources in Sweave. In this case the un-Sweaved files can be read in any text editor. This offers the blind user an opportunity to learn the necessary commands required to complete tasks.
8. The batch mode for running R jobs provides an alternative means of directing the input and output to files that can be viewed in the user's preferred applications. A common tool for gaining access to software that has difficulty interacting with screen readers is to direct output to a text file which can be viewed in a web browser. As the file is updated via the interaction with the statistical software, the text file is altered. Using the refresh functionality of the browser means the user can continue to interact with the statistical software even though it is not accessible. This is possible using R, but not necessary. It is a useful way of running medium sized tasks that might have to be tweaked once or twice. Certainly, this author uses batch mode in situations where his colleagues use R interactively.
9. R offers the user access to the information that feeds into a graph in a way that is not paralleled in most statistical software. Whenever a user creates a graph, R processes the data into the form needed to create that summary graph. The object is often stored (implicitly) and discarded once the graph is formed. Some functions print this information out at the user's bidding (a histogram for example where the user has set `plot=FALSE` as an argument). Wrapping a `print` command around a graph command in R creates the graph and prints out the object used to create it. Occasionally this object is not desirable but there are situations where it provides access to the graph that is not otherwise possible.

Making base R usable by blind useRs

For users of the Windows operating system, the terminal window is the best way to use R interactively. This means the user must alter the shortcut placed on the desktop as part of the standard installation. This is all the blind user must do which makes the task of getting a new software application ready for use with a screen reader. Aside from doing nothing as software is ready to go, this is about as easy as one could ever expect. Many software solutions have special options for screen readers hidden away in the menus somewhere and the worst of these would require a sighted person to alter the settings on the blind user's behalf.

This author has used this approach without issue under Windows XP, but use of R in the terminal window of Windows Vista and Windows 7 has proved problematic for the screen reading software which can "lose focus". This means the cursor is no longer able to be controlled by the blind user and the session must be curtailed. This problem is not specific to a particular screen reader, and can be replicated in other terminal windows such as the command prompt of these operating systems. At the time of writing, this problem remains and the author can only recommend using base R in batch mode to compensate.

The option of using the R console under Windows in conjunction with a screen reader is feasible. The greatest benefit the terminal window offers is that the results from commands are voiced by the screen reader automatically. In the console mode, the screen reader must be switched into a mode that reviews the screen's contents. If the output is longer than can fit on screen, the blind user will not be able to access some of the output.

Those blind people working with Macintosh or Linux systems need to do very little. The Mac user must decide if they will use the terminal window or rely on the console version, but either is fine. Linux users will be working with a terminal-like environment so will be working with R straight out of the box. Both of these operating systems have screen reader software built in. The VoiceOver product for Apple products has clear sound and can be controlled easily. The ORCA screen reader for Linux is not quite as sophisticated and can be more difficult to use. The blind people using these operating systems have done so by choice and know how to work with these screen readers. Less experienced blind people tend to use the Windows operating system, often with a commercial screen reader, although freeware options exist.

In late July 2011, the author exposed a number of blind and low vision students to the opportunities R has to offer them in their upcoming studies as a back-stop should their chosen university's software preference prove inaccessible. These blind students were able to use R with a variety of screen readers (with and without braille displays) under the three major operating systems, including one Linux user who used a braille display without the addition of the screen reader's speech output. The participants were given accessible documentation that guided them through use of basic R commands for creating

simple data objects, creating simple graphs using the **datasets** package, and (optionally) to create simple regression or ANOVA models.

How could R be better?

Given the superiority of R for use by blind people over its competitors, there is a certain reluctance to propose issues. In most cases, however, I think the developers of R and its many users would appreciate gaining an understanding of its limitations for the blind users they may have to work with in the future.

First and most obviously is the inability of a blind person to use the excellent R Commander package **Rcmdr** (Fox, 2005) and similar front ends. The fault is not with the front end itself, but is due to the inability of the toolkit used to create the front end to communicate with the screen reader using each operating system's native API. Unfortunately, the Tcl/Tk GUI toolkit, accessed using the **tcltk** package in R, does not communicate with the screen reading software used by blind people to drive the speech output or braille displays. It is possible to teach the screen reader to engage properly with applications built using the Tcl/Tk GUI toolkit, but every change made in the front end needs to be taught to the screen reader. To the author's knowledge, no implementation of the Tcl/Tk toolkit has created accessible front ends for any software so R is not unusual in this respect. The blind user wishing to use packages like R Commander will therefore only benefit from the functions offered as command line tools.

In direct response to a referee's request, I have investigated the usefulness of RStudio (RStudio, 2013) to a blind user. My findings are that this front end is not useful to a blind user relying on screen reading software. As a consequence of the referee's request, I have created a web page[1] that details my experiences with R, especially when using such tools as R Commander and RStudio. I welcome opportunities to test developments in the wider R Project that may benefit blind users and update this resource accordingly.

There is no question that a requirement for developers of front ends to think of the needs of blind users is unreasonable as we are such a small audience. It does need to be pointed out however that the selection of the toolkit itself is where fault lies, if any is to be apportioned. Personally, I do not feel inclined to find fault, but if a successor to R comes to fruition, then I would want the toolkit included to be one that engages with every operating system in such a way that additional screen reader support is not required. The wxWidgets library (Smart et al., 2011) is an example of a toolkit for developing a GUI that creates applications that do communicate with screen readers without special adjustments being made by the developers or those that have an interest in screen reader software. Such toolkits are unfortunately seldom used and perhaps the most relevant example worth noting is the front end developed for Maxima (Maxima.sourceforge.net, 2012) called wxMaxima[2].

One of the benefits of using a GUI creation toolkit that communicates with the screen reading software is that accessibility features are generally built into those toolkits and little or no extra work is required from the developer. For example, most GUI applications have hot keys on menu items. Once a user (blind or sighted) learns the hot keys for commonly used menu items they tend to use them. Some of these hot keys are able to be used directly from the command line, while those found most commonly in R are key letters for menu items that can be selected once the menu is on screen. It should be noted that the provision of hot keys does not guarantee access or efficiency. The blind user still needs to remember the key combinations required to avoid slow browsing or having to remember the location of menu items.

A feature that would be useful for blind users, and perhaps many others, is the desire to save the console (or terminal) window including all input and output for future reference. For the blind user this is an essential means of copying content to a document as they cannot use the mouse to highlight the elements desired. The work around is to use the `sink` command, but this stores output only. Operating in batch mode is the only real option unless additional packages are used that create this functionality, such as that found in the **TeachingDemos** package (Snow, 2010), or the `Sweave` functionality found in the **utils** package.

The use of HTML for the help pages in R has been mentioned as a major drawcard for the blind user. I can report that low vision users would prefer to have greater control over the way these pages are displayed so they can make the best use of their residual vision. This may include the use of colour as well as different font families and styles.

The use of PDF as the format of choice for vignettes and other documentation creates some difficulty for many blind users. The PDF format is not as easy to work with for many reasons, with the total

[1] See `http://r-resources.massey.ac.nz/StatSoftware` for more details.

[2] The accessible front end wxMaxima has only recently featured as a built-in component of Maxima. See `http://maxima.sourceforge.net/` for details and to download this software.

inaccessibility of equations and mathematical notation being a key example. Creation of HTML, or preferably XML in place of PDF for vignettes and documentation would improve the accessibility of this material. If XML is used in place of HTML, the blind reader can use the MathPlayer[3] plug in for Internet Explorer which converts the mathematical expression into a plain English text string (Soiffer, 2005). This approach is satisfactory for the vast majority of formulae encountered in an introductory statistics course. Failing this, the blind user must wait until such endeavours as Moore (2009) to improve the accessibility of PDF documents created by LaTeX/TeX are successful. Access to equations in PDF documents is a further challenge which is only just receiving attention[4].

There are several ways to create either HTML or XML documents from LaTeX, notably using the TeX4ht[5] package (Gurari, 2004). As an alternative, the Plastex[6] package is used to create the online documentation (presented in HTML not PDF) for SAS (SAS Institute Inc, 2011).

A second issue with reading PDF files is that many blind users will not be able to deal with the ligatures used in the basic LaTeX fonts. These are the letter combinations that are printed as a single character, such as 'fi', 'ff', or 'ft'. It is possible to disable ligatures in many instances; the competent user may add the necessary LaTeX commands when they re-process the files distributed with add-on packages. Creation of these files in HTML is therefore also an option available to the blind user, but is probably not an option for the novice R/LaTeX user.

A final element of R that is not currently accessible to blind users is the data window. This spreadsheet screen for editing/browsing a data frame directly has proven to be of little value to the blind user under Windows. Fortunately, the blind user can learn how to use R commands to investigate their raw data, or can fall back on other accessible spreadsheet software to do this.

Introducing BrailleR

The **BrailleR** package is under development (Godfrey, 2012) and an initial offering appears on CRAN, as well as having its own homepage[7]. In broad terms the package aims to provide blind users a text interpretation of graph objects, using the (implicitly) stored list object created as part of the function called.

This is being implemented by creation of a method called the `VI` method as a first step. It is working for some simple graph types such as histograms. Use of a method is reliant on assignment of a distinct class attribute by the function being used to create the graph. Assignment of class attributes to more objects would broaden the scope of the `VI` method.

In situations where the class attribute is not yet assigned, creation of a wrapper function that will do the same job is possible. This back-up plan is seen as an undesirable but pragmatic means of achieving the aims of the package. Any functionality worked on in this group of functions will be transferred to the method once the corresponding class attribute assignment is assured.

The functionality offered by the `VI` method is aimed at describing what the sighted graph user can see as fact, without forming an opinion. This means the user is then reliant on interpreting what the graph tells the sighted person. In the case of a histogram, the `VI` method will detail the total number of items, the midpoints of bars, and the counts of objects in each bin. The shape of the distribution is left to the user to determine. The method is therefore not meant as any form of expert system. Readers can compare the output from base R functionality with that offered via **BrailleR** for a histogram of a sample of random normal values using:

```
require("BrailleR")
x <- rnorm(1000)
hist(x, plot=FALSE)
VI(hist(x))
```

Displaying summary statistics for data frames is a common task that leads to a nicely formatted display for the sighted person, but a screen reader reads the details line by line. This means hearing all the minimum values, then the lower quartiles, etc. and is often just too difficult to contemplate as a blind user. Simplifying the process of creating summary statistics for data frames is an example of where the blind person's efficiency can be improved by **BrailleR**.

[3] MathPlayer is available from http://www.dessci.com/mathplayer/ and is a free download.

[4] Neil Soiffer of Design Science is working on this problem courtesy of a grant received from the Institute of Education Sciences (U.S. Department of Education) in 2012.

[5] More details about TeX4ht are available at http://tug.org/applications/tex4ht/mn.html

[6] Plastex, found at http://plastex.sourceforge.net/ (Smith, 2009), is an open source project for converting LaTeX into HTML and other formats.

[7] The **BrailleR** project home page is at http://r-resources.massey.ac.nz/BrailleR.

Another efficiency can be brought about by developing functions that fit specific models to a data set, e.g., a multiple regression. The blind user (and sighted ones too) might find it useful to have a function that fits the model and automatically saves the associated output in a text file, as well as placing the residual plots in their own files. It makes sense to automate the process of creating and saving graphs given that the blind user will not actually look at the graphs, and may wish to insert them into a document at a later time. Alternative approaches for considering residual analyses are a key task for implementation in **BrailleR**. This may need to be done using numerical approaches, rules of thumb, and alternative hypothesis testing such as direct consideration of a Levene's test for homogeneity in an analysis of variance example.

Plans for interpreting more difficult constructs such as scatter plots are being considered. The main issue for finding a suitable interpretation is that of simplicity. How does one explain a scatter plot simply and effectively, without the benefit of statistical ideas to someone that cannot see the graph? Further, how would one do this when encountering such a graph in an examination context without answering the question for the blind student? This is not easy. The author has benefitted from overly keen examination supervisors in the past — sometimes he has been limited by their ability too.

In order to meet the needs of a blind user wishing to review an entire R session (warts and all), the terminal or console window needs to be saved in a text file. Functionality for this was found in the **TeachingDemos** package, and the **BrailleR** package is the better for the ability to make use of this functionality directly. Thanks are due to Greg Snow for the code and help files which are now included in **BrailleR**. Users wishing to use a more sophisticated file type than plain text are directed to the use of the **R2HTML** package (Lecoutre, 2003), although the author prefers to use the `Sweave` functionality of the **utils** package.

Anyone who has experience working with blind students using R, or any other statistical software is invited to contact the author with details of their experiences and resulting questions. Your enquiries will help improve the **BrailleR** package.

Summary

Yes, R could be better, but at the current time it is the best software for blind statisticians. This article has documented the fact that R is accessible to blind users of all operating systems. It has suggested some improvements that will make R even better and therefore set the standard for all statistical software in terms of meeting the needs of blind people around the world.

Running R in the terminal window instead of using the GUI mode of operation means the blind (Windows) user has instant access to as much functionality as their sighted peers. For Windows users, this means a simple alteration to the default shortcut placed on the desktop. The only improvement over this would be if the standard console window was accessible for screen reading software. Given the vast development already undertaken to extend R using the useful but inaccessible Tcl/Tk toolkit, it is likely that the blind community will need to continue operating using the terminal mode.

Of course, it is hoped that any successor to R will incorporate the most recent accessibility standards at the outset and that it will not lose the advantages R currently offers the blind community.

Bibliography

J. Ebert. Scientists with disabilities: Access all areas. *Nature*, 435(7042):552–554, 2005. [p73]

J. Fox. The R Commander: A basic statistics graphical user interface to R. *Journal of Statistical Software*, 14(9):1–42, 2005. URL http://www.jstatsoft.org/v14/i09. [p76]

J. Gardner, C. Herden, G. Herden, C. Dreyer, and G. Bulatova. Simultaneous Braille tactile graphics and ink with Tiger ink attachment and Duxbury. In *Proceedings of the 2004 CSUN International Conference on Technology and Persons with Disabilities*, Los Angeles, California, Mar. 2004. [p73]

A. J. R. Godfrey. *BrailleR: Improved access for blind useRs*, 2012. R package version 0.4. [p77]

E. M. Gurari. TEX4ht: HTML production. *TUGboat*, 25(1):39–47, 2004. URL http://www.tug.org/TUGboat/tb25-1/gurari.pdf. [p77]

E. Lecoutre. The R2HTML package. *R News*, 3(3):33–36, Dec. 2003. URL http://cran.r-project.org/doc/Rnews/Rnews_2003-3.pdf. [p78]

Maxima.sourceforge.net. *Maxima, a Computer Algebra System*, 2012. URL http://maxima.sourceforge.net/. [p76]

R. Moore. Ongoing efforts to generate "tagged PDF" using pdfTEX. *TUGboat*, 30(2):170–175, 2009. URL http://www.tug.org/TUGboat/tb30-2/tb95moore.pdf. [p77]

RStudio. *RStudio: Integrated Development Environment for R, Version 0.97.336*. RStudio, Boston, MA, 2013. URL http://www.rstudio.com/. [p76]

SAS Institute Inc. *The SAS System, Version 9.3.* SAS Institute Inc, Cary, NC, 2011. URL http://www.sas.com/. [p77]

J. Smart, R. Roebling, V. Zeitlin, R. Dunn, et al. *wxWidgets 2.8.12: A portable C++ and Python GUI toolkit*, Mar. 2011. URL http://docs.wxwidgets.org/stable/. Accessed on 2012-02-20. [p76]

K. D. Smith. *Plastex, A Python Framework for Processing LaTeX Documents*, 2009. URL http://plastex.sourceforge.net/. [p77]

G. Snow. *TeachingDemos: Demonstrations for teaching and learning*, 2010. URL http://CRAN.R-project.org/package=TeachingDemos. R package version 2.7. [p76]

N. Soiffer. MathPlayer: Web-based math accessibility. In *Proceedings of the 7th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 204–205, 2005. [p77]

T. Watanabe, T. Yamaguchi, and M. Nakagawa. Development of tactile graph automated creation software. In K. Yamaguchi and M. Suzuki, editors, *Proceedings of Digitization and E-Inclusion in Mathematics and Science*, pages 143–146, Tokyo, Japan, 2012. [p73]

A. Jonathan R. Godfrey
Massey University
Palmerston North
New Zealand
a.j.godfrey@massey.ac.nz

PIN: Measuring Asymmetric Information in Financial Markets with R

by Paolo Zagaglia

Abstract The package **PIN** computes a measure of asymmetric information in financial markets, the so-called probability of informed trading. This is obtained from a sequential trade model and is used to study the determinants of an asset price. Since the probability of informed trading depends on the number of buy- and sell-initiated trades during a trading day, this paper discusses the entire modelling cycle, from data handling to the computation of the probability of informed trading and the estimation of parameters for the underlying theoretical model.

Introduction

A large set of finance packages are available in R for an array of tasks including, among others, portfolio analysis (e.g., see Boudt et al., 2010; Kane et al., 2011). Only a small set of tools is available for the study of intra-daily transactions, mostly focused on data-management issues. Yet these represent only the first building block for modelling the dynamics of asset prices.

In recent years, the rise of algorithmic and high-frequency trading has shed a bright light on the fact that movements in financial market prices are largely affected by microstructure forces. In particular, some traders in the market may enjoy privileged information about the value of an asset, while others may be trading merely on the basis of public news. The risk that an individual *uninformed* trader may face an *informed* counterparty in the market-place is then an important determinant of asset prices (e.g., see Easley et al., 2002). Hence, measuring the probability that a counterparty enjoys asymmetric information, also known as *probability of informed trading* (PIN), allows a trader to correctly price a security (see Easley and O'Hara, 1987).

The **PIN** package (Zagaglia, 2013) provides tools for calculating the probability of informed trading proposed by Easley et al. (1996). This measure is obtained by estimating a model of strategic interaction between traders with different information sets. In particular, the probability of informed trading depends on the number of buy- and sell-trades taking place in the market.

Knowledge about whether a trade is initiated by the buying or selling party is not always available. In this case, the so-called *signing* of trades can be obtained from applying alternative classification algorithms on trade and quote data, when available (e.g., see Lee and Ready, 1991). These functionalities are already offered by excellent packages in R.

To illustrate the complementarity between these data-handling tools and the **PIN** package, I present a sample application that uses the **highfrequency** package of Cornelissen et al. (2013). I consider a data set of fictitious trades and quotes for a given trading day that is well representative of standard limited order books. This allows me to consider a preliminary issue surrounding the availability of data. Quotes often need to be matched to the available trades, in the sense of making the time stamps of the two consistent with each other. Only after this is done, can the user compute the number of implicit buy- and sell-trades, estimate the parameters of the sequential trade model of Easley et al. (1996), and compute the resulting PIN. In other words, I discuss the entire empirical modelling cycle for the probability of informed trading.

I should stress that typical applications of the PIN focus on the stock market. This is mostly due to the large availability of accessible data. However, recent empirical contributions have estimated the PIN for alternative markets, such as the U.S. government bond market (see Li et al., 2009). Idier and Nardelli (2011) provide an application to the European overnight money market. This is an over-the-counter market, for which only representative price quotes are available. Estimating the probability of informed trading on markets that do not feature a transparent organization of exchanges—for instance, in the form of an order book or a market-maker—requires adapting the model of Easley et al. (1996) to take into account the relevant idiosyncratic features. For this reason, the discussion in this paper focuses on standard market structures, such as those of today's stock markets, and disregards the more controversial case of over-the-counter markets.

R packages for market microstructure analysis

The basic prerequisite for any application in market microstructure is the availability of proper financial data. Hence, the R developers' community has proposed a number of efficient and powerful tools to address the resulting big-data challenge.

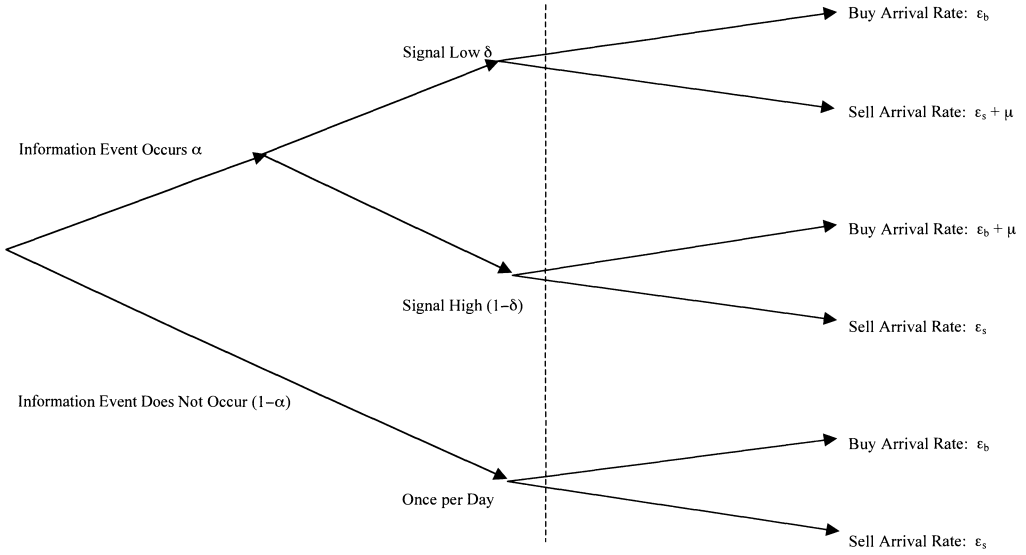


Figure 1: The trading game

The starting point consists in downloading the data set from a provider. The package **IBrokers** by Ryan (2011) provides access to the data-feed supplied by Interactive Brokers and automates data requests for the creation of high-frequency data sets.

Given the availability of a financial-market data set, a variety of excellent functions are available in R for data management, including data re-shuffling, plotting and computing descriptive statistics. Kane et al. (2011) presents the package **orderbook**, which provides data-handling functionalities from limited order books. Boudt and Cornelissen (2012) present R functions for managing the trade and quote (TAQ) data set from NASDAQ and NYSE in R. TAQ data set files are available in different forms that report different types of information, potentially involving a detailed structure of the order books. This can easily result in terabytes of data. For this purpose, Emerson and Kane have extended the available packages with the aim of handling terabyte-large data sets[1].

Only a few tools are available to the user community for the study of market-microstructure issues in R. Enos et al. (2008) present a library for the computation of trade costs in market transaction of stocks. In a recent conference presentation[2], Yan and Zivot discuss the implementation of models for price impact and discovery in the stock market.

The **RTAQ** package has recently been incorporated into the **highfrequency** package (Cornelissen et al., 2013). In this extension, the user is provided with functions to compute different measures of market liquidity, such as those of Hasbrouck and Seppi (2001) and Venkataraman (2001). This package also provides functionalities for classifying trade origination. Not all the markets provide public information on whether a trade starts either from the buy-side, or from the sell-side. This is a prerequisite for studying a large number of topics in market microstructure, including asymmetric information and informed trading. For this purpose, several methods have been proposed to infer trade direction from changes in market prices and quotes. The **highfrequency** package implements an effective modification of the so-called *tick rule* proposed by Lee and Ready (1991).

Computing the probability of informed trading

The theoretical model

Easley et al. (1996) build on the sequential trading model of Glosten and Milgrom (1987) and Kyle (1985) to estimate the probability that a counterpart in the trading process enjoys superior information. This is based on the assumption that the market is composed by a heterogeneous population of traders. Some traders are *informed*, while others are *uninformed*.

Within each trading day, a competitive market-maker buys or sells a security at the bid and ask

[1] Towards Terabytes of TAQ, R in Finance 2012, http://www.rinfinance.com/agenda/2012/talk/Emerson+Kane.pdf

[2] Estimating the Dynamics of Price Discovery, R in Finance 2012, http://www.rinfinance.com/agenda/2012/talk/Yan+Zivot.pptx

quotes that he or she announce publicly to the market. Since the market-maker is risk-neutral and competes with the traders in the transactions, the resulting prices represent the expected value of the stock conditional on the market-maker's information at the time of trade.

Before the beginning of a trading day, nature determines whether an information event relevant to the value of the stock will occur. Information events are independently distributed and occur with probability α. Good news events take place with a probability δ and bad news events occur with a probability $1-\delta$. At the end of the day, the information set of each actor in the market is complete and the true value of the stock is realized.

The market-maker knows both the probability of these events taking place and the arrival order of information to the market. However, he or she does not know which event is realized. Easley et al. (1996) assume that the market-maker is Bayesian in that they use the information from trade arrival to update their belief about information events. Since the information set is independent across days, the model structure allows the change in beliefs to be considered separately on each day.

In this framework, the market is made by the traders that can observe the signals about the true stock value and that benefit from asymmetric information. There are also uninformed traders that receive no news about the stock price. These two groups of traders enter the market with a frequency determined by independent Poisson processes at any minute within a trading day.

An information-revealing event causes informed traders to enter the market. For instance, if a 'high-value' or 'good' signal materializes, these traders buy the stock. The arrival of news events both to the market and to each trader follow independent Poisson processes.

The set of events and associated probabilities can be represented in the trading tree of Figure ??. At the first node of the tree, nature selects whether an information event occurs. If an event does occur, nature determines the sign of the resulting news. The three nodes of 'no event', 'good news' and 'bad news' placed before the dotted line of Figure ?? occur only once per day. Given the node selected for the day, traders arrive according to the relevant Poisson processes. On good-event days, the arrival rates are $\epsilon+\mu$ for buy orders and ϵ for sell orders. On bad-event days, the arrival rates are ϵ for buys and $\epsilon+\mu$ for sells. Finally, on days without information-revealing news, only uninformed traders take part in the market with an arrival rate of ϵ.

While observing changes in the order book, traders obtain information about order arrival. However, they cannot observe the underlying motives for trading by counterparties. In other words, traders do not know the parameter values of the model consistent with trading patterns. However, these parameters can be estimated from data on order arrivals.

Empirical implementation of the theoretical model

In order to reduce the parameter space, it is typically assumed that the sell- and buy-arrival rates ϵ_b and ϵ_s are equal. On a good-day event, the probability of observing a sequence of buy- and sell-initiated trades is given by

$$e^{-(\mu+\epsilon)T}\frac{[(\mu+\epsilon)T]^B}{B!}e^{\epsilon T}\frac{(\epsilon T)^S}{S!} \tag{1}$$

On a bad-signal day, this probability becomes

$$e^{\epsilon T}\frac{(\epsilon T)^B}{B!}e^{-(\mu+\epsilon)T}\frac{[(\mu+\epsilon)T]^S}{S!} \tag{2}$$

Finally, when information-revealing events do not take place, the likelihood of observing orders is equal to

$$e^{\epsilon T}\frac{(\epsilon T)^B}{B!}e^{\epsilon T}\frac{(\epsilon T)^S}{S!} \tag{3}$$

Assuming that trading activity is independent across T days, the likelihood of trading activity takes the form

$$\begin{aligned}\mathrm{L}[\{B,S\}|\theta] = {} & (1-\alpha)e^{-\epsilon T}\frac{(\epsilon T)^B}{B!}e^{-\epsilon T}\frac{(\epsilon T)^S}{S!} \\ & + \alpha\delta e^{-\epsilon T}\frac{(\epsilon T)^B}{B!}e^{-(\mu+\epsilon)T}\frac{[(\mu+\epsilon)T]^S}{S!} \\ & + \alpha(1-\delta)e^{-(\mu+\epsilon)T}\frac{[(\mu+\epsilon)T]^B}{B!}e^{-\epsilon T}\frac{(\epsilon T)^S}{S!}\end{aligned} \tag{4}$$

with the parameter space $\theta=\alpha,\delta,\epsilon,\mu$. Over h independent days, the likelihood of observing the data

$M = (B_i, S_i)_{i=1}^{h}$ is the product of the daily likelihoods

$$\mathrm{L}\left[M|\theta\right] = \prod_{i=1}^{h} \mathrm{L}\left(\theta|B_i, S_i\right) \tag{5}$$

To deal with the issue of convergence in numerical maximization, I follow Aktasa et al. (2007) and Easley et al. (2008), and re-arrange the likelihood function 5 as:

$$\begin{aligned}\mathrm{L}\left[M|\theta\right] = \sum_{t=1}^{T} \left[-2\epsilon + M_t \ln x + (B_t + S_t)\ln(\mu+\epsilon)\right] \\ + \sum_{t=1}^{T} \ln\left[\alpha(1-\alpha)e^{-mu}x^{S_t - M_t} + \alpha\delta e^{-\mu}x^{B_t - M_t} + (1-\alpha)x^{B_t + S_t - M_t}\right]\end{aligned} \tag{6}$$

with $M_t = \min(B_t, S_t) + \max(B_t, S_t)/2$, and $x_t = \epsilon/(\mu+\epsilon)$.

Consistent with the literature (e.g., see Easley et al., 2002, 1996), the role of asymmetric information is measured by the probability of informed trading (PIN_t). This is defined as the unconditional probability that informed traders buy or sell an asset at each point in time, and is equal to

$$\mathrm{PIN}_t = \frac{\alpha\mu}{\alpha\mu + 2\epsilon} \tag{7}$$

When this probability is high, uninformed traders face a higher risk of trading with a counterparty that is better informed.

The package PIN includes an R function for the computation of the log-likelihood function 6. This takes as inputs the parameter values—in the order $\epsilon, \mu, \alpha, \delta$—and the time series of daily data on number of buy- and sell-initiated trades—arranged as a $n \times w$ matrix, where n is the number of trading days. The first column of this data matrix includes the number of buy trades and the second one has the number of sell trades.

A practical example

In this section I present an example of how to integrate the PIN package with other tools available in R for handling market-microstructure data sets. In particular, the application presented here relies on the capabilities provided by the **highfrequency** package.

I discuss the computation of PIN using a data set of fictitious quotes and trades for an ideal stock. These quotes and trades are provided as two data sets–`sample_qdata` and `sample_tdata`, respectively—in the public distribution of the package **highfrequency**.[3]

The noteworthy characteristic of the database for trades and quotes is that its structure follows the TAQ format for standard NYSE stocks. In other words, the approach discussed here is general enough to handle a data source that has been a key reference in the industry.

It should be stressed that the fictitious trades and quotes refer to one trading day only. This is an evident simplification, which is typically avoided in empirical studies of PIN. For instance, Easley et al. (2002) exclude stocks that have less than 60 days of quotes and trades data, suggesting that would disrupt the reliability of the model parameter estimates. The PIN package allows estimations over any sample length to be run. Moreover, the application presented in this paper can easily be extended to cover more than one trading day.

As a starting point, I load the relevant packages into R, including those required by the **highfrequency** package:

```
library(zoo)
library(xts)
library(highfrequency)
library(TTR)
library(timeDate)
library(quantmod)
library(PIN)
```

Then, I prepare the data set for the estimation of PIN. The **highfrequency** package includes two sample files with the trades and quotes. These two pieces of information need to be matched, in the sense of ordering the quoting activity that is consistent with the recorded trades within a given trading day:

[3]Hence, data for these fictitious trades and quotes are available online to the R community.

```
#Load data samples
data(sample_tdata)
data(sample_qdata)

#Match the trade and quote data
tqdata <- matchTradesQuotes(sample_tdata, sample_qdata)
```

The data set used in this example does not include information on trade direction. As stressed earlier, this is an issue that is present in certain markets and can be addressed using the trade classification algorithms proposed in the finance literature. Here I use the functionality of the **highfrequency** package to apply the Lee-Ready approach that combines trading and quoting activity:

```
x <- getTradeDirection(tqdata)

tradeDirection <- matrix(x)

buy_side <- which(tradeDirection > 0)

num_buy_side <- length(matrix(buy_side))
num_sell_side <- length(tradeDirection) - length(matrix(buy_side))

ntrades <- cbind(num_buy_side, num_sell_side)
```

The maximization of the log-likelihood function 6 is a prerequisite for the computation of the PIN. Ideally, one should perform a constrained optimization and restrict the spaces of all the parameters over the unit interval $[0, 1]$. For this particular data set, I employ the general optimization routine `optim`:

```
initparams <- cbind(0.15, 0.05, 0.5, 0.5)

options(warn = -1)
param_optim <- optim(initparams, pin_likelihood, gr = NULL, ntrades)
```

The optimizing parameters are 1, 0.3715623, 0.7255069 and 0.2808594. These estimated values suggest that there is a high probability for the trading day to reveal information (i.e., $\hat{\alpha} \approx 0.72$. When there are no news events, only uninformed traders take part in the market ($\epsilon = 1$).

Finally, given the optimizing parameters, the computation of the probability of informed trading is straightforward:

```
epsi <- param_optim$par[1]
miu  <- param_optim$par[2]
alph <- param_optim$par[3]
delt <- param_optim$par[4]

pin <- (alph*miu)/(alph*miu + 2*epsi)
```

In this case, I obtain a low probability of trading with an informed counterparty, which is equal to 0.1187762.

Conclusion

This article discusses an application in R of concepts and methods of financial market microstructure. In particular, I present a package for the computation of the probability of informed trading. This is a standard measure of asymmetric information that is based on trade direction. I show how the proposed PIN package fits into the broader context of the available tools for handling microstructure data.

The routines outlined here could be extended along several dimensions. It would be useful to include alternative measures of information asymmetry. The time-varying PIN specification of Easley et al. (2008) represents a relevant candidate. Also, the Volume-Synchronized Probability of Informed Trading - VPIN - is a key information metric that is increasingly becoming the industry standard (see Easley et al., 2012).

The provision of liquidity to the market is related to the presence of asymmetric information. This consideration is especially relevant for market-makers, who manage inventories against the risk that counterparties have superior information on the true value of an asset. Hence, any tool that computes the PIN provides an incomplete picture of the market if it says nothing about the sources for

market liquidity. For this reason, it would be beneficial to provide tools for the measurement of price pressures, such as those proposed by Hendershott and Menkveld (1991) for the NYSE.

Acknowledgements

The author is very grateful to two anonymous referees for very effective comments that have shaped the revision of this article to a major extent.

Bibliography

N. Aktasa, E. de Bodta, F. Declerckc, and H. Van Oppensa. The PIN Anomaly around M&A Announcements. *Journal of Financial Markets*, 10(2):169–191, 2007. [p83]

K. Boudt and J. Cornelissen. *RTAQ*, 2012. URL http://cran.r-project.org/src/contrib/Archive/RTAQ/. R package version 0.2. [p81]

K. Boudt, P. Carl, and B. G. Peterson. *PortfolioAnalytics: Portfolio Analysis, including Numeric Methods for Optimization of Portfolios*, 2010. URL http://r-forge.r-project.org/projects/returnanalytics/. R package version 0.6. [p80]

J. Cornelissen, K. Boudt, and S. Payseur. *highfrequency*, 2013. URL http://CRAN.R-project.org/package=highfrequency. R package version 0.2. [p80, 81]

D. Easley and M. O'Hara. Price, Trade Size, and Information in Securities Markets. *Journal of Financial Economics*, 19(1):69–90, 1987. [p80]

D. Easley, N. M. Kiefer, M. O'Hara, and J. B. Paperman. Liquidity, Information, and Infrequently Traded Stocks. *Journal of Finance*, 51(4):1405–1436, 1996. [p80, 81, 82, 83]

D. Easley, S. Hvidkjaer, and M. O'Hara. Is Information Risk a Determinant of Asset Returns? *Journal of Finance*, 57(5):2185–2221, 2002. [p80, 83]

D. Easley, R. F. Engle, M. O'Hara, and L. Wu. Time-Varying Arrival Rates of Informed and Uninformed Traders. *Journal of Financial Econometrics*, 6(2):171–207, 2008. [p83, 84]

D. Easley, M. M. L. de Prado, and M. O'Hara. Flow Toxicity and Liquidity in a High Frequency World. Working paper, 2012. [p84]

J. Enos, D. Kane, A. R. Narayan, A. Schwartz, D. Suo, and L. Zhao. Trade Costs. *R News*, 8(1):10–13, 2008. [p81]

L. R. Glosten and P. R. Milgrom. Bid Ask and Transaction Price in a Specialist Market with Heterogeneously Informed Traders. *Journal of Financial Economics*, 14(1):71–100, 1987. [p81]

J. Hasbrouck and D. J. Seppi. Common Factors in Prices, Order Flows and Liquidity. *Journal of Financial Economics*, 59(1):383–411, 2001. [p81]

T. Hendershott and A. J. Menkveld. Price Pressures. Unpublished version, 209, 1991. [p85]

J. Idier and S. Nardelli. Probability of Informed Trading on the Euro Overnight Market Rate. *International Journal of Finance and Economics*, 16(2):131–145, 2011. [p80]

D. Kane, A. Liu, and K. Nguyen. Analyzing an Electronic Limit Order Book. *The R Journal*, 2(64–68):1, 2011. [p80, 81]

A. S. Kyle. Continuous Auctions and Insider Trading. *Econometrica*, 53(6):1315–1335, 1985. [p81]

C. M. C. Lee and M. J. Ready. Inferring Trade Direction from Intraday Data. *Journal of Finance*, 46(2):733–746, 1991. [p80, 81]

H. Li, J. Wang, C. Wu, and Y. He. Are Liquidity and Information Risks Priced in the Treasury Bond Market? *Journal of Finance*, 64(1):467–503, 2009. [p80]

J. A. Ryan. *IBrokers*, 2011. URL http://CRAN.R-project.org/package=IBrokers/. R package version 0.9-10. [p81]

K. Venkataraman. Automated Versus Floor Trading: An Analysis of Execution Costs on the Paris and New York Exchanges. *Journal of Finance*, 56(2):1445–1485, 2001. [p81]

P. Zagaglia. *PIN: Estimates the Parameters of a Trading-tree Model for the Computation of the Probability of Informed Trading*, 2013. URL http://CRAN.R-project.org/package=PIN. R package version 0.8. [p80]

Paolo Zagaglia
International Entrepreneurship Academy
Dublin, Ireland
Rimini Centre for Economic Analysis and
Department of Economics and Department of Cultural Goods, Università di Bologna
Bologna, Italy
paolo.zagaglia@gmail.com

QCA: A Package for Qualitative Comparative Analysis

by Alrik Thiem and Adrian Duşa

Abstract We present **QCA**, a package for performing Qualitative Comparative Analysis (QCA). QCA is becoming increasingly popular with social scientists, but none of the existing software alternatives covers the full range of core procedures. This gap is now filled by **QCA**. After a mapping of the method's diffusion, we introduce some of the package's main capabilities, including the calibration of crisp and fuzzy sets, the analysis of necessity relations, the construction of truth tables and the derivation of complex, parsimonious and intermediate solutions.

Introduction

Qualitative Comparative Analysis (QCA) - a research method popularized largely through the work of Charles Ragin (Ragin, 1987, 2000, 2008) - counts among the most influential recent innovations in social science methodology. In line with Ragin's own background, QCA has been initially employed only by a small number of (political) sociologists (e.g., Amenta et al., 1992; Griffin et al., 1991; Wickham-Crowley, 1991). Since then, however, the method has made inroads into political science and international relations (e.g., Thiem, 2011; Vis, 2009), business and economics (e.g., Evans and Aligica, 2008; Valliere et al., 2008), management and organization (e.g., Greckhamer, 2011), legal studies and criminology (Arvind and Stirton, 2010; Miethe and Drass, 1999), education (e.g., Glaesser and Cooper, 2011; Schneider and Sadowski, 2010), and health policy research (e.g., Harkreader and Imershein, 1999; Schensul et al., 2010). Figure 1 charts the trend in the total number of QCA applications that have appeared in peer-reviewed journal articles since 1984, broken down by its three variants *crisp-set QCA* (csQCA), *multi-value QCA* (mvQCA) and *fuzzy-set QCA* (fsQCA).[1]

Allowing for a publication lag of about two years, 4.2 applications on average have been published throughout the first decade following the introduction of csQCA in Ragin (1987). But only his sequel "Fuzzy-Set Social Science" (Ragin, 2000) seems to have got the "Ragin Revolution" (Vaisey, 2009) eventually off ground. In the years from 2003 to 2007, the average number of applications had risen to 13.6 before the absolute number of applications more than tripled from 12 in 2007 to 39 in 2011. Despite the introduction of fsQCA, applications of csQCA have continued to increase from four in 2001 to 22 in 2011. In contrast to csQCA and fsQCA, mvQCA has remained underutilized. Of a total of 280 applications between 1984 and 2012, only ten have employed this variant. Even when accounting for the fact that it has been introduced in 2004, 17 years after csQCA and four years after fsQCA, this represents a disproportionately low number.[2]

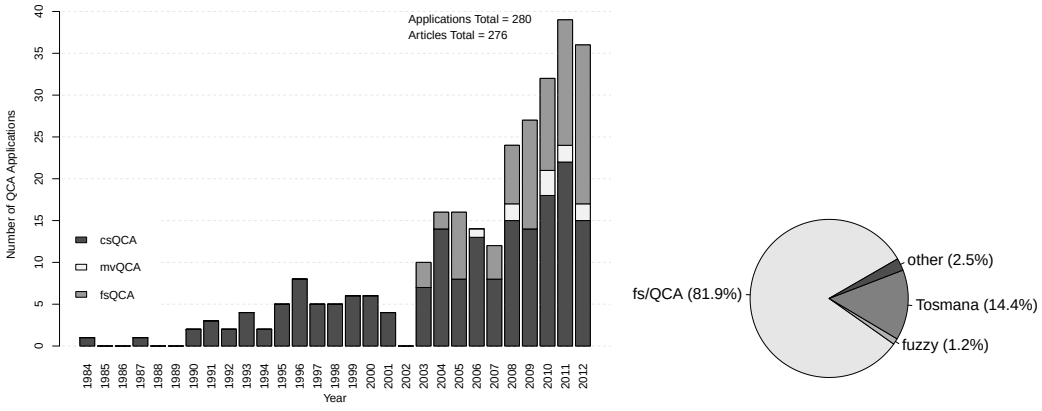


Figure 1: Trend in number of QCA applications by variants and year, and software market share.

QCA's methodological success story has created a growing demand for tailored software, which

[1]The number of applications differs slightly from the number of articles as four articles have each presented two applications of QCA using two different variants. In order to be included in the data, applications had to focus primarily on a substantive research question, not QCA as a method. All entries have been recorded in the bibliography section on `http://www.compasss.org`.

[2]See the debate in *Field Methods* for more details on the status of mvQCA (Thiem, 2013; Vink and van Vliet, 2009, 2013).

has been met almost exclusively by two programmes: Charles Ragin and Sean Davey's (2009) fs/QCA and Lasse Cronqvist's (2011) Tosmana. Until recently, however, users of non-Windows operating systems were limited as neither programme ran on other operating systems than Microsoft Windows. As of version 1.3.2.0, Tosmana has also supported other operating systems. In 2008 and 2012, Kyle Longest and Stephen Vaisey's (2008) fuzzy package for Stata and Christopher Reichert and Claude Rubinson's (2013) Kirq have been developed as alternatives to fs/QCA. For the R environment, Adrian Duşa's **QCA** package has been first added in 2006 and in 2009, Ronggui Huang (2011) has released the **QCA3** package. The detailed market shares of these software solutions are also shown in Figure 1.[3] Holding a clear monopoly, fs/QCA is by far the most common software with 82%, followed by Tosmana with 14% and fuzzy with 1%. Other solutions have claimed about 3%, but neither R package has managed to win any market shares thus far.

Not as unequal as their market shares, but significantly different still, are the capabilities of these software solutions. Table 1 provides an overview of the functionality each programme offers. All alternatives to **QCA** have different capabilities, but none covers the entire range of basic procedures. Kirq, fs/QCA and fuzzy cannot handle multi-value sets, whereas Tosmana cannot process fuzzy sets. The possibility to analyze necessity relations is not implemented in Tosmana, either, and the other packages, except Kirq, offer only limited user-driven routines. Complex and parsimonious solutions can be found by all packages, but only fs/QCA generates intermediate solutions on an automated basis.

Table 1: Overview Software Functionality [a]

Function	Tosmana[b]	Kirq[c]	fs/QCA[d]	fuzzy[e]	QCA3[f]	QCA[g]
			variant			
csQCA	●	●	●	●	●	●
mvQCA	●	○	○	○	●	●
fsQCA	○	●	●	●	●	●
(tQCA)	○	○	●	○	●	●
			solution type			
complex	●	●	●	●	●	●
intermediate	○	◐	●	○	◐	●
parsimonious	●	●	●	●	●	●
			procedure			
necessity tests	○	●	◐	◐	◐	●
parameters of fit	○	●	●	◐	◐	●
calibration	◐	○	◐	◐	◐	●
factorization	○	○	○	○	○	●
identify (C)SAs	●	○	○	○	●	●
statistical tests	○	○	○	●	◐	○

[a] ○/ ◐/ ● = no/partial/full functionality

[b] version 1.3.2.0; [c] version 2.1.9; [d] version 2.5; [e] version st0140_2; [f] version 0.0-5; [g] version 1.0-5

The calibration of crisp sets is limited in fs/QCA and **QCA3**. Tosmana cannot handle fuzzy sets, but it provides more elaborate tools for the calibration of crisp sets. In addition to Ragin's (2008) "direct method" and "indirect method", fuzzy offers a set-normalizing linear membership function. Most importantly, it also includes various statistical procedures for coding truth tables, the appropriateness of which largely depends on the research design.[4]

QCA combines and enhances the individual strengths of other software solutions. It can process all QCA variants (including temporal QCA (tQCA)), generates all solution types, and offers a wide range of procedures. For example, **QCA** provides four classes of functions for almost all kinds of calibration requirements and has an automated routine for the analysis of necessity relations. **QCA** is also the only package that can factorize any Boolean expression. As in Tosmana and **QCA3**, *simplifying assumptions* can also be identified. Unlike Tosmana, however, which does not present any parameters of fit, **QCA** produces inclusion, coverage and PRI (Proportional Reduction in Inconsistency) scores for

[3] Only 156 out of 276 articles cite the software program that was used.

[4] As QCA has been initially developed for medium-*n* (≈10-30 cases) macro-level data analysis, often with universes of objects rather than random samples, statistical tests are not only often meaningless but also rarely produce statistically significant configurations. With large-*n* micro-level data, such tests may be more appropriate.

both necessity and sufficiency relations in mvQCA.[5]

In summary, a comprehensive QCA software solution has been missing so far. Researchers have often been limited in their analyses when using one programme, or they had to resort to different programmes for performing all required operations. This gap is now filled by the **QCA** package, which seeks to provide a user-friendly yet powerful command-line interface alternative to the two dominant graphical user interface solutions fs/QCA and Tosmana. In the remainder of this article, we introduce some of the package's most important functions, including the calibration of sets, the analysis of necessity relations, the construction of truth tables and the derivation of complex, parsimonious and intermediate solution types.

Calibration of sets

The process of translating *base variables* (also referred to as *raw data*) into condition or outcome set membership scores is called *calibration*, in fsQCA also *fuzzification*. In contrast to csQCA, continuous base variables need not be categorized directly in fsQCA but can be transformed with the help of continuous functions, a procedure called *assignment by transformation* (Verkuilen, 2005, p. 465). Ragin (2008), for example, suggests a piecewise-defined logistic function. Sufficient for the vast majority of fuzzification needs, **QCA** offers the `calibrate()` function, one of whose flexible function classes for positive end-point concepts is given in Equation (1).

$$\mu_{\mathbf{S}}(b) = \begin{cases} 0 & \text{if } \tau_{\text{ex}} \geq b, \\ \frac{1}{2}\left(\frac{\tau_{\text{ex}} - b}{\tau_{\text{ex}} - \tau_{\text{cr}}}\right)^p & \text{if } \tau_{\text{ex}} < b \leq \tau_{\text{cr}}, \\ 1 - \frac{1}{2}\left(\frac{\tau_{\text{in}} - b}{\tau_{\text{in}} - \tau_{\text{cr}}}\right)^q & \text{if } \tau_{\text{cr}} < b \leq \tau_{\text{in}}, \\ 1 & \text{if } \tau_{\text{in}} < b. \end{cases} \tag{1}$$

Here, b is the base variable, τ_{ex} the threshold for full exclusion from set $\mathbf{S}$, τ_{cr} the crossover threshold at the point of maximally ambiguous membership in $\mathbf{S}$ and τ_{in} the threshold for full inclusion in $\mathbf{S}$. The parameters p and q control the degrees of concentration and dilation. The piecewise-defined logistic membership function suggested in Ragin (2008) is also available. Furthermore, `calibrate()` can generate set membership scores for sets based on negative or positive mid-point concepts (Thiem and Duşa, 2013, pp. 55-62). If no suitable thresholds have been found even after all means of external and internal identification have been exhausted, **QCA**'s `findTh()` function can be employed for searching thresholds using hierarchical cluster analysis.

```
> library(QCA)
> # base variable and vector of thresholds
> b <- sort(rnorm(15)); th <- quantile(b, c(0.1, 0.5, 0.9))

> # create bivalent crisp set
> calibrate(b, thresholds = th[2])
 [1] 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

> # create trivalent crisp set using thresholds derived from cluster analysis
> calibrate(b, thresholds = findTh(b, groups = 3))
 [1] 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2

> # fuzzification using Equation (1)
> round(calibrate(b, type = "fuzzy", thresholds = th), 2)
 [1] 0.00 0.00 0.04 0.32 0.42 0.47 0.48 0.50 0.59 0.72 0.77 0.93 0.94 1.00 1.00

> # negation of previous result
> round(calibrate(b, type = "fuzzy", thresholds = rev(th)), 2)
 [1] 1.00 1.00 0.96 0.68 0.58 0.53 0.52 0.50 0.41 0.28 0.23 0.07 0.06 0.00 0.00

> # fuzzification using piecewise logistic
> round(calibrate(b, type = "fuzzy", thresholds = th, logistic = TRUE), 2)
 [1] 0.02 0.04 0.06 0.25 0.38 0.45 0.46 0.50 0.64 0.79 0.83 0.93 0.93 0.96 0.97
```

[5] We prefer the term "inclusion" over the more common term "consistency" as introduced by Ragin (2006) because the measure represents the degree to which one set is included by another, producing evidence that is either consistent with the underlying hypothesis, inconsistent or mixed. The measure is also called Inclusion-1 index (cf. Smithson and Verkuilen, 2006, p. 65).

Analysis of necessity

Whenever the occurrence of an event B is accompanied by the occurrence of an event A, then B implies A ($B \Rightarrow A$) and A is implied by B ($A \Leftarrow B$). Put differently, A is necessary for B and B is sufficient for A. Transposed to the set-theoretic terminology of QCA, analyses of necessity proceed from the observation of a *value* under the *outcome set* **Y** - written $\mathbf{Y}\{v_l\}$ - to the observation of a value under the *condition set* **X** - written $\mathbf{X}\{v_l\}$. For analyzing *necessity inclusion*, the decisive question is to which degree objects are members of $\mathbf{X}\{v_l\}$ and $\mathbf{Y}\{v_l\}$ in relation to their overall membership in $\mathbf{Y}\{v_l\}$. If necessity inclusion is high enough, the evidence is consistent with the hypothesis that $\mathbf{X}\{v_l\}$ is necessary for $\mathbf{Y}\{v_l\}$ ($\mathbf{X}\{v_l\} \supseteq \mathbf{Y}\{v_l\}$). The formula for necessity inclusion $\text{Incl}_N(\mathbf{X}\{v_l\})$ is presented in Equation (2).

$$\text{Incl}_N(\mathbf{X}\{v_l\}) = \frac{\sum_{i=1}^{n} \min(\{v_l\}x_i, \{v_l\}y_i)}{\sum_{i=1}^{n} \{v_l\}y_i} \tag{2}$$

Provided that $\mathbf{X}\{v_l\} \supseteq \mathbf{Y}\{v_l\}$ is sufficiently true, *necessity coverage* allows an assessment of the frequency with which B occurs relative to A. The formula for necessity coverage $\text{Cov}_N(\mathbf{X}\{v_l\})$ is given in Equation (3).

$$\text{Cov}_N(\mathbf{X}\{v_l\}) = \frac{\sum_{i=1}^{n} \min(\{v_l\}x_i, \{v_l\}y_i)}{\sum_{i=1}^{n} \{v_l\}x_i} \tag{3}$$

For analyzing necessity relations, **QCA** offers the `superSubset()` function. If p_j denotes the number of values of condition set j with $j = 1, 2, \ldots, k$, the function returns necessity inclusion, PRI and coverage scores for those of the $d = \prod_{j=1}^{k}(p_j + 1) - 1$ combinations of condition set values that just meet the given inclusion and coverage cut-offs.[6] Therefore, `superSubset()` does not require users to predefine the combinations to be tested, and so removes the risk of leaving potentially interesting results undiscovered. The initial set of combinations always consists of all $\prod_{j=1}^{k} p_j$ trivial intersections $\langle \mathbf{X}_1\{v_1\}, \mathbf{X}_1\{v_2\}, \ldots, \mathbf{X}_1\{v_p\}, \ldots, \mathbf{X}_k\{v_p\} \rangle$. The size of the intersection is incrementally increased from 1 to k until its inclusion score falls below the cut-off. If no trivial intersection passes the inclusion cut-off, `superSubset()` automatically switches to forming set unions until the least complex form has been found.

For demonstration purposes, we reanalyze the data from Krook's (2010) csQCA on women's representation in 22 Western democratic parliaments. Countries with electoral systems of proportional representation (**ES**), parliamentary quotas (**QU**), social democratic welfare systems (**WS**), autonomous women's movements (**WM**), more than 7% left party seats (**LP**) and more than 30% seats held by women (**WNP**) are coded "1", all others "0". The first five sets are the conditions to be tested for necessity in relation to the outcome set **WNP**. For reasons of simplicity and space, we use lower case letters for denoting set negation in all remaining code examples.

```
> data(Krook)
> Krook
   ES QU WS WM LP WNP
SE  1  1  1  0  0   1
FI  1  0  1  0  0   1
NO  1  1  1  1  1   1
..  .  .  .  .  .   .
<rest omitted>

> superSubset(Krook, outcome = "WNP", cov.cut = 0.52)

             incl   PRI    cov.r
--------------------------------
1  ES+LP     1.000  1.000  0.733
2  ES+WM     1.000  1.000  0.524
3  WS+WM+LP  1.000  1.000  0.611
4  QU+wm+LP  1.000  1.000  0.550
5  QU+WM+lp  1.000  1.000  0.524
6  QU+WS+LP  1.000  1.000  0.550
7  QU+WS+WM  1.000  1.000  0.524
8  es+QU+WS  1.000  1.000  0.524
--------------------------------
```

[6] The inclusion cut-off always enjoys priority over the coverage cut-off.

When not specified otherwise, all sets in the data but the outcome are assumed to be conditions. By default, the function tests for necessity, but sufficiency relations can also be analyzed. No trivial intersection has passed the inclusion cut-off and `superSubset()` has thus formed unions of conditions. Substantively, the first combination **ES** + **LP** means that having proportional representation or strong left party representation is necessary for having more than 30% parliamentary seats held by women.

Analysis of sufficiency, step 1: Truth tables

Whenever the occurrence of an event A is accompanied by the occurrence of an event B, then A implies B ($A \Rightarrow B$) and B is implied by A ($B \Leftarrow A$). Put differently, A is sufficient for B and B is necessary for A. Transposed to the set-theoretic terminology of QCA, analyses of sufficiency proceed from the observation of a value under **X** to the observation of a value under **Y**. For analyzing *sufficiency inclusion*, the decisive question is to which degree objects are members of $\mathbf{X}\{v_l\}$ and $\mathbf{Y}\{v_l\}$ in relation to their overall membership in $\mathbf{X}\{v_l\}$. If sufficiency inclusion is high enough, the evidence is consistent with the hypothesis that $\mathbf{X}\{v_l\}$ is sufficient for $\mathbf{Y}\{v_l\}$ ($\mathbf{X}\{v_l\} \subseteq \mathbf{Y}\{v_l\}$). The formula for sufficiency inclusion $\text{Incl}_S(\mathbf{X}\{v_l\})$ is presented in Equation (4).

$$\text{Incl}_S(\mathbf{X}\{v_l\}) = \frac{\sum_{i=1}^{n} \min(\{v_l\}x_i, \{v_l\}y_i)}{\sum_{i=1}^{n} \{v_l\}x_i} \tag{4}$$

The classical device for analyzing sufficiency relations is the *truth table*, which lists all $d = \prod_{j=1}^{k} p_j$ *configurations* and their corresponding outcome value.[7] Configurations represent exhaustive combinations of set values characterizing the objects. For illustration, a simple hypothetical truth table with three bivalent condition sets $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$ and the outcome value OUT is presented in Table 2. Three bivalent conditions yield the eight configurations listed under $\mathcal{C}_i$. The minimum number of cases n that is usually required for the respective outcome value is also appended.

Table 2: Hypothetical Truth Table

$\mathcal{C}_i$	$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$	OUT	n
1	1	1	1	1	≥ 1
2	1	1	0	1	≥ 1
3	1	0	1	1	≥ 1
4	1	0	0	1	≥ 1
5	0	1	1	0	≥ 1
6	0	1	0	C	≥ 2
7	0	0	1	?	0
8	0	0	0	?	0

It is important to emphasize that the outcome value is not the same as the outcome set, the latter of which does not show up in QCA truth table. Instead, the *outcome value* is based on the sufficiency inclusion score, returning a truth value that indicates the degree to which the evidence is consistent with the hypothesis that a sufficiency relation between a configuration and the outcome set exists. Configurations $\mathcal{C}_1$ to $\mathcal{C}_4$ are *positive* because they support this hypothesis (OUT = 1), $\mathcal{C}_5$ is *negative* because it does not (OUT = 0). Mixed evidence exists for $\mathcal{C}_6$ (OUT = C). If at least two objects conform to one configuration, but the evidence neither sufficiently confirms nor falsifies the hypothesis of a subset relation between this configuration and the outcome set, *contradictions* arise. No empirical evidence at all exists for $\mathcal{C}_7$ and $\mathcal{C}_8$. If a configuration has no or too few cases, it is called a *logical remainder* (OUT = ?).

The `truthTable()` function can generate truth tables for all three main QCA variants without users having to specify which variant they use. The structure of the data is automatically transposed into the correct format.

```
> KrookTT <- truthTable(Krook, outcome = "WNP")
> KrookTT

 OUT: outcome value
   n: number of cases in configuration
```

[7] This table is also often called "table of combinations" (McCluskey, 1965) or "function table" (Hohn, 1966).

```
incl: sufficiency inclusion score
 PRI: proportional reduction in inconsistency

    ES QU WS WM LP OUT n  incl  PRI
 3  0  0  0  1  0   0  2  0.000 0.000
 4  0  0  0  1  1   1  1  1.000 1.000
 9  0  1  0  0  0   0  1  0.000 0.000
11  0  1  0  1  0   0  4  0.000 0.000
12  0  1  0  1  1   1  1  1.000 1.000
18  1  0  0  0  1   0  1  0.000 0.000
21  1  0  1  0  0   1  1  1.000 1.000
24  1  0  1  1  1   1  1  1.000 1.000
25  1  1  0  0  0   0  3  0.000 0.000
26  1  1  0  0  1   1  1  1.000 1.000
27  1  1  0  1  0   1  1  1.000 1.000
28  1  1  0  1  1   1  2  1.000 1.000
29  1  1  1  0  0   1  1  1.000 1.000
32  1  1  1  1  1   1  2  1.000 1.000
```

At a minimum, `truthTable()` requires an appropriate dataset and the `outcome` argument, which identifies the outcome set. If `conditions` is not provided as an argument, it is assumed that all other sets in the data but the outcome are the conditions. By default, logical remainders are not printed unless specified otherwise by the logical argument `complete`. The logical argument `show.cases` prints the names of the objects next to the configuration in which they have membership above 0.5.

The `truthTable()` function includes three cut-off arguments that influence how `OUT` is coded. These are `n.cut`, `incl.cut1` and `incl.cut0`. The first argument `n.cut` sets the minimum number of cases with membership above 0.5 needed in order to not code a configuration as a logical remainder. The second argument `incl.cut1` specifies the minimal sufficiency inclusion score for a non-remainder configuration to be coded as positive. The third argument `incl.cut0` offers the possibility of coding configurations as contradictions when their inclusion score is neither high enough to consider them as positive nor low enough to code them as negative. If the inclusion score of a non-remainder configuration falls below `incl.cut0`, this configuration is always considered negative. By means of the `sort.by` argument, the truth table can also be ordered along inclusion scores, numbers of cases or both, in increasing or decreasing order. If the original condition set labels are rather long, the logical `letters` argument can be used to replace the set labels with upper case letters in alphabetical order.

The leftmost column list the configuration row index values from the complete truth table. Sufficiency inclusion and PRI scores are also provided in the two rightmost columns. Once the truth table is fully coded, it can be minimized according to the theorems of Boolean algebra (McCluskey, 1965, pp. 84-89).

Analysis of sufficiency, step 2: Boolean minimization

The *canonical union* resulting from the truth table presented in Table 2 is given by Equation (5). It consists of four *fundamental intersections* (FI), each of which corresponds to one positive configuration. Generally, all FIs also represent positive configurations, but not all positive configurations become FIs. The analyst may decide to exclude some of these configurations from the minimization process on theoretical or empirical grounds.

$$\overbrace{\mathbf{X}_1 \cap \mathbf{X}_2 \cap \mathbf{X}_3}^{\mathcal{C}_1} \cup \overbrace{\mathbf{X}_1 \cap \mathbf{X}_2 \cap \mathbf{x}_3}^{\mathcal{C}_2} \cup \overbrace{\mathbf{X}_1 \cap \mathbf{x}_2 \cap \mathbf{X}_3}^{\mathcal{C}_3} \cup \overbrace{\mathbf{X}_1 \cap \mathbf{x}_2 \cap \mathbf{x}_3}^{\mathcal{C}_4} \subseteq \mathbf{Y} \tag{5}$$

If two FIs differ on the values of one condition only, then this condition can be eliminated so that a simpler term results. For example, Equation (5) can be reduced in two passes as shown in Figure 2. In the first pass, the four FIs can be reduced to four simpler terms. In the second pass, these four terms can then be reduced at once to a single term. No further reduction is possible, $\mathbf{X}_1$ is the only term which is essential with respect to the outcome ($\mathbf{X}_1 \subseteq \mathbf{Y}$). All terms that survive the Boolean minimization process are called *prime implicants* (PI).

The central function of the **QCA** package that performs the minimization is `eqmcc()` (enhanced Quine-McCluskey) (Duşa, 2007, 2010). It can derive *complex*, *parsimonious* and *intermediate solutions* from a truth table object or a suitable dataset. In contrast to complex solutions, parsimonious solutions incorporate logical remainders into the minimization process without any prior assessment by the analyst as to whether a sufficiency relation is plausible or not. Intermediate solutions offer a middle way insofar as those logical remainders that have been used in the derivation of the parsimonious

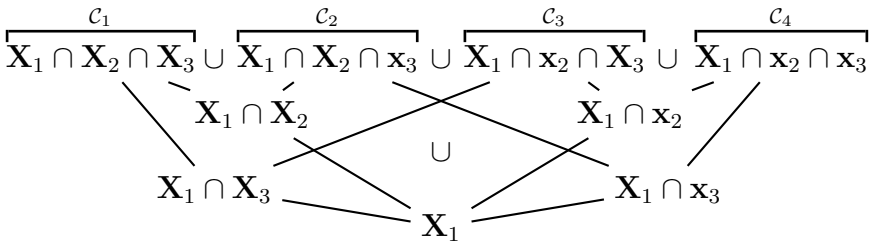


Figure 2: Boolean minimization of Equation (5).

solution are filtered according to the analyst's *directional expectations* about the impact of each single condition set value on the overall sufficiency relation of the configuration of which it is part and the outcome set. By formulating such expectations, *difficult logical remainders* are excluded as FIs from the canonical union, whereas those logical remainders that enter the canonical union are *easy*. The complex solution, which is the default option, can be generated by `eqmcc()` with minimal typing effort.

```
> KrookSC <- eqmcc(KrookTT, details = TRUE)
> KrookSC

n OUT = 1/0/C: 11/11/0
  Total      : 22

S1: ES*QU*ws*LP + ES*QU*ws*WM + es*ws*WM*LP + ES*WS*wm*lp + ES*WS*WM*LP

             incl   PRI    cov.r  cov.u
----------------------------------------
ES*QU*ws*LP  1.000  1.000  0.273  0.091
ES*QU*ws*WM  1.000  1.000  0.273  0.091
es*ws*WM*LP  1.000  1.000  0.182  0.182
ES*WS*wm*lp  1.000  1.000  0.182  0.182
ES*WS*WM*LP  1.000  1.000  0.273  0.273
----------------------------------------
S1           1.000  1.000  1.000
```

The truth table object `KrookTT` that was generated above is passed to `eqmcc()`. No further information is necessary in order to arrive at the complex solution. The logical argument `details` causes all parameters of fit to be printed together with the *minimal union* `S1`: inclusion (`incl`), PRI (`PRI`), raw coverage (`cov.r`) and unique coverage (`cov.u`) scores for each PI as well as the minimal union.[8] If `details = TRUE`, the logical argument `show.cases` also prints the names of the objects that are covered by each PI.

If alternative minimal unions exist, all of them are printed if the row dominance principle for PIs is not applied as specified in the logical argument `rowdom`. One PI $\mathcal{P}_1$ dominates another $\mathcal{P}_2$ if all FIs covered by $\mathcal{P}_2$ are also covered by $\mathcal{P}_1$ and both are not interchangeable (cf. McCluskey, 1965, p. 150). Inessential PIs are listed in brackets in the solution output and at the end of the PI part in the parameters-of-fit table, together with their unique coverage scores under each individual minimal union. For example, the parsimonious solution without row dominance applied can be derived by making all logical remainders available for inclusion in the canonical union as FIs and by setting `rowdom` to `FALSE`.

```
> KrookSP <- eqmcc(KrookTT, include = "?", rowdom = FALSE, details = TRUE)
> KrookSP

n OUT = 1/0/C: 11/11/0
  Total      : 22

S1: WS + ES*WM + QU*LP + (es*LP)
S2: WS + ES*WM + QU*LP + (WM*LP)

                          -------------------
      incl   PRI    cov.r  cov.u  (S1)   (S2)
```

[8] Unique coverage scores do not apply to minimal unions.

```
-------------------------------------------
WS     1.000  1.000  0.455  0.182  0.182  0.182
ES*WM  1.000  1.000  0.545  0.091  0.091  0.091
QU*LP  1.000  1.000  0.545  0.091  0.091  0.091
-------------------------------------------
es*LP  1.000  1.000  0.182  0.000  0.091
WM*LP  1.000  1.000  0.636  0.000         0.091
-------------------------------------------
S1     1.000  1.000  1.000
S2     1.000  1.000  1.000
```

The intermediate solution for bivalent set data requires a vector of directional expectations in the `direxp` argument, where "0" denotes absence, "1" presence and "-1" neither. The intermediate solution with all conditions expected to contribute to a positive outcome value when present is generated as follows:

```
> KrookSI <- eqmcc(KrookTT, include = "?", direxp = c(1,1,1,1,1), details = TRUE)
> KrookSI

n OUT = 1/0/C: 11/11/0
  Total      : 22

p.sol: WS + ES*WM + QU*LP + WM*LP

S1:    ES*WS + WM*LP + ES*QU*LP + ES*QU*WM

          incl   PRI    cov.r  cov.u
-----------------------------------
ES*WS     1.000  1.000  0.455  0.182
WM*LP     1.000  1.000  0.636  0.182
ES*QU*LP  1.000  1.000  0.455  0.091
ES*QU*WM  1.000  1.000  0.455  0.091
-----------------------------------
S1        1.000  1.000  1.000
```

For intermediate solutions, `eqmcc()` also prints the parsimonious solution (`p.sol`) whose simplifying assumptions have been used in filtering logical remainders. The PI chart of this intermediate solution (`i.sol`) that has been derived from the (first and only) complex and the (first and only) parsimonious solution (`C1P1`) can then be inspected by accessing the corresponding component in the returned object.

```
> KrookSI$PIchart$i.sol$C1P1

          4  12 21 24 26 27 28 29 32
ES*WS     -  -  x  x  -  -  -  x  x
WM*LP     x  x  -  x  -  -  x  -  x
ES*QU*LP  -  -  -  -  x  -  x  -  x
ES*QU*WM  -  -  -  -  -  x  x  -  x
```

If several minimal sums exist under both the parsimonious and complex solution, the PI chart of the respective combination for the intermediate solution can be accessed by replacing the numbers in the `C1P1` component.

Besides the PI chart, the solution object returned by `eqmcc()` also contains a dataframe of PI set membership scores in the `pims` component. These scores can then be used to draw Venn diagrams of solutions, similar to the one shown in Figure 3, using suitable R packages such as **VennDiagram** (Chen and Boutros, 2011).

```
> KrookSI$pims$i.sol$C1P1
   ES*WS WM*LP ES*QU*LP ES*QU*WM
SE     1     0        0        0
FI     1     0        0        0
NO     1     1        1        1
DK     1     1        0        0
NL     0     1        1        1
ES     0     0        0        1
```

```
..     .     .        .        .
<rest omitted>
```

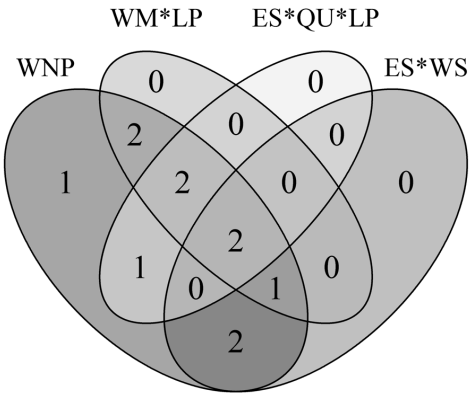


Figure 3: Venn diagram of three PIs from intermediate solution produced with the **VennDiagram** package.

Summary

In recent years, Qualitative Comparative Analysis (QCA) has become the method of choice for testing configurational hypotheses. However, QCA is still very much a "method in the making". Extensions, enhancements and alternative algorithms appear on a regular basis (Baumgartner, 2009; Eliason and Stryker, 2009; Schneider and Wagemann, 2012; Thiem, 2012). R provides an ideal environment within which established QCA procedures as well as more advanced techniques can be implemented in a manner as transparent and user-responsive as possible. The **QCA** package makes a significant contribution in this regard. It fills the individual gaps in other programs' coverage of basic functionality, and provides further improvements through complementary and advanced procedures.

The QCA software market remains dominated by the two graphical user interface programmes fs/QCA and Tosmana. **QCA** seeks to bridge the method of QCA with powerful command-line software while retaining a user-friendly code and command structure. In order to lower the barriers for social scientists to choosing R for QCA further, we have published an introductory textbook with extended replications of recent QCA studies from various research areas (Thiem and Duşa, 2013). Although most examples have been taken from political science, the book may also be of interest to researchers from related disciples. At the same time, this textbook also serves as a comprehensive reference manual for the **QCA** package.

Bibliography

E. Amenta, B. G. Carruthers, and Y. Zylan. A hero for the aged: The Townsend movement, the political mediation model, and United-States old-age policy, 1934-1950. *American Journal of Sociology*, 98(2): 308–339, 1992. [p87]

T. T. Arvind and L. Stirton. Explaining the reception of the Code Napoleon in Germany: A fuzzy-set qualitative comparative analysis. *Legal Studies*, 30(1):1–29, 2010. [p87]

M. Baumgartner. Inferring causal complexity. *Sociological Methods & Research*, 38(1):71–101, 2009. [p95]

H. Chen and P. Boutros. Venndiagram: A package for the generation of highly-customizable Venn and Euler diagrams in R. *BMC Bioinformatics*, 12(1):35–41, 2011. [p94]

L. Cronqvist. *Tosmana, version 1.3.2.0 [computer program]*. University of Trier, Trier, 2011. [p88]

A. Duşa. *Enhancing Quine-McCluskey*. WP 2007-49, COMPASSS, 2007. URL http://www.compasss.org/files/WPfiles/Dusa2007a.pdf. [p92]

A. Duşa. A mathematical approach to the Boolean minimization problem. *Quality & Quantity*, 44(1): 99–113, 2010. [p92]

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. [p95]

A. J. Evans and P. D. Aligica. The spread of the flat tax in Eastern Europe. *Eastern European Economics*, 46(3):49–67, 2008. [p87]

J. Glaesser and B. Cooper. Selectivity and flexibility in the German secondary school system: A configurational analysis of recent data from the German socio-economic panel. *European Sociological Review*, 27(5):570–585, 2011. [p87]

T. Greckhamer. Cross-cultural differences in compensation level and inequality across occupations: A set-theoretic analysis. *Organization Studies*, 32(1):85–115, 2011. [p87]

L. J. Griffin, C. Botsko, A.-M. Wahl, and L. W. Isaac. Theoretical generality, case particularity: Qualitative comparative analysis of trade-union growth and decline. *International Journal of Comparative Sociology*, 32(1-2):110–136, 1991. [p87]

S. Harkreader and A. W. Imershein. The conditions for state action in Florida's health-care market. *Journal of Health and Social Behavior*, 40(2):159–174, 1999. [p87]

F. E. Hohn. *Applied Boolean algebra: An elementary introduction*. Macmillan, New York, 2nd edition, 1966. [p91]

R. Huang. *QCA3: Yet another package for qualitative comparative analysis*, 2011. R package version 0.0-4. [p88]

M. L. Krook. Women's representation in parliament: A qualitative comparative analysis. *Political Studies*, 58(5):886–908, 2010. [p90]

K. C. Longest and S. Vaisey. fuzzy: A program for performing qualitative comparative analyses (QCA) in Stata. *Stata Journal*, 8(1):79–104, 2008. [p88]

E. J. McCluskey. *Introduction to the theory of switching circuits*. Princeton University Press, Princeton, 1965. [p91, 92, 93]

T. D. Miethe and K. A. Drass. Exploring the social context of instrumental and expressive homicides: An application of qualitative comparative analysis. *Journal of Quantitative Criminology*, 15(1):1–21, 1999. [p87]

C. C. Ragin. *The comparative method: Moving beyond qualitative and quantitative strategies*. University of California Press, Berkeley, 1987. [p87]

C. C. Ragin. *Fuzzy-set social science*. University of Chicago Press, Chicago, 2000. [p87]

C. C. Ragin. Set relations in social research: Evaluating their consistency and coverage. *Political Analysis*, 14(3):291–310, 2006. [p89]

C. C. Ragin. *Redesigning social inquiry: Fuzzy sets and beyond*. University of Chicago Press, Chicago, 2008. [p87, 88, 89]

C. C. Ragin and S. Davey. *fs/QCA, version 2.5 [computer program]*. Department of Sociology, University of Arizona, Tucson, AZ, 2009. [p88]

C. Reichert and C. Rubinson. *Kirq, version 2.1.9 [computer program]*. University of Houston-Downtown, Houston, TX, 2013. [p88]

J. Schensul, D. Chandran, S. Singh, M. Berg, S. Singh, and K. Gupta. The use of qualitative comparative analysis for critical event research in alcohol and HIV in Mumbai, India. *AIDS and Behavior*, 14(S1): 113–125, 2010. [p87]

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. [p95]

P. Schneider and D. Sadowski. The impact of new public management instruments on PhD education. *Higher Education*, 59(5):543–565, 2010. [p87]

M. Smithson and J. Verkuilen. *Fuzzy set theory: Applications in the social sciences*. SAGE, London, 2006. [p89]

A. Thiem. Conditions of intergovernmental armaments cooperation in Western Europe, 1996-2006. *European Political Science Review*, 3(1):1–33, 2011. [p87]

A. Thiem. *Unifying configurational comparative methodology: Generalized-set qualitative comparative analysis*. Working Paper Series, 2012-34, IPSA Committee on Concepts & Methods, 2012. URL http://www.concepts-methods.org. [p95]

A. Thiem. Clearly crisp, and not fuzzy: A reassessment of the (putative) pitfalls of multi-value QCA. *Field Methods*, 25(2):197–207, 2013. [p87]

A. Thiem and A. Duşa. *Qualitative comparative analysis with R: A user's guide*. Springer, New York, 2013. [p89, 95]

S. Vaisey. QCA 3.0: The "Ragin revolution" continues. *Contemporary Sociology: A Journal of Reviews*, 38 (4):308–312, 2009. [p87]

D. Valliere, N. Na, and S. Wise. Prior relationships and M&A exit valuations: A set-theoretic approach. *Journal of Private Equity*, 11(2):60–72, 2008. [p87]

J. Verkuilen. Assigning membership in a fuzzy set analysis. *Sociological Methods & Research*, 33(4): 462–496, 2005. [p89]

M. P. Vink and O. van Vliet. Not quite crisp, not yet fuzzy? Assessing the potentials and pitfalls of multi-value QCA. *Field Methods*, 21(3):265–289, 2009. [p87]

M. P. Vink and O. van Vliet. Potentials and pitfalls of multi-value QCA: Response to Thiem. *Field Methods*, 25(2):208–213, 2013. [p87]

B. Vis. Governments and unpopular social policy reform: Biting the bullet or steering clear? *European Journal of Political Research*, 48(1):31–57, 2009. [p87]

T. P. Wickham-Crowley. A qualitative comparative approach to Latin American revolutions. *International Journal of Comparative Sociology*, 32(1-2):82–109, 1991. [p87]

Alrik Thiem
Department of Humanities, Social and Political Sciences
Swiss Federal Institute of Technology Zurich (ETHZ)
IFW C29.2, Haldeneggsteig 4
8092 Zurich
Switzerland
thiem@sipo.gess.ethz.ch

Adrian Duşa
Department of Sociology
University of Bucharest
1 Schitu Măgureanu
050025 Bucharest
Romania
dusadrian@unibuc.ro

An Introduction to the EcoTroph R Package: Analyzing Aquatic Ecosystem Trophic Networks

by Mathieu Colléter, Jérôme Guitton and Didier Gascuel

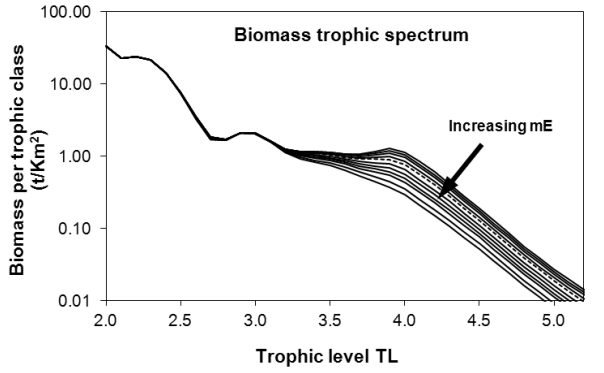
Abstract Recent advances in aquatic ecosystem modelling have particularly focused on trophic network analysis through trophodynamic models. We present here a R package devoted to a recently developed model, **EcoTroph**. This model enables the analysis of aquatic ecological networks and the related impacts of fisheries. It was available through a plug-in in the well-known Ecopath with Ecosim software or through implementations in Excel sheets. The R package we developed simplifies the access to the EcoTroph model and offers a new interfacing between two widely used software, Ecopath and R.

Introduction

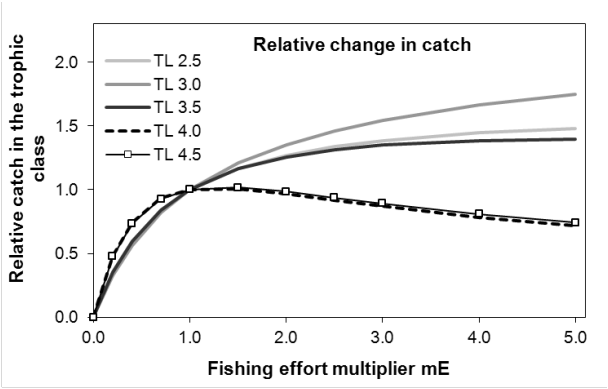
In the face of the global overexploitation of marine resources and the fast degradation of ecosystems integrity, scientists developed new modelling approaches at the scale of the ecosystem. In fact, the main tool used for fisheries regulation is a stock approach which does not account for the trophic network linking marine ecological components. An important challenge is to analyze the aquatic ecological networks and the related impacts of fishery. There are several ways to define and represent an ecosystem. One widely used approach is based on the trophic level concept. Elton (1927) and Lindeman (1942) introduced this concept for describing aquatic ecosystems by assigning integer trophic levels (TLs) to the individual numbers, to the biomass or to the biological production by its component species. This approach differentiated between primary producers and detritus (TL = 1), first-order consumers (TL = 2), second-order consumers (TL = 3) ... The ecosystem is so represented as a pyramid of number, biomass or production, from low to high TLs. Odum and Heald (1975) developed this concept by implementing fractional trophic levels resulting from the diet of the individual and the trophic level of its preys. The emergence of Ecopath as a widely used approach and software for modelling aquatic ecosystems (Polovina, 1984; Christensen and Pauly, 1992) contributed in a major way to the prominence of TLs, especially as they were not an input, but an output of the model (i.e. estimated parameters). As the use of Ecopath spread worldwide with hundreds of application cases, so did the trophic level concept.

EcoTroph (ET) is an approach and software for modelling marine and freshwater ecosystems, entirely articulated around the TL concept (Gascuel, 2005; Gascuel and Pauly, 2009). It has been developed at the same time as the Ecopath worldwide expansion happened and incorporated into the Ecopath plug-in family (Gascuel et al., 2009). The first key idea of ET is that it deals with the continuous distribution of the biomass in an ecosystem as a function of continuous TL. The biomass enters the foodweb at TL = 1, generated by the photosynthetic activity of primary producers, or recycled from the detritus by the microbial loop. Between TL = 1 and TL = 2, the biomass is composed of mixotrophs only, and is usually low. If any, it is conventionally split between biomasses at TL = 1 and 2. Then, at TLs $\geq$ 2, the biomass is composed by heterotrophic organisms with mixed diet and fractional TLs resulting in a continuous distribution of biomass along TLs. The second key feature of ET is that the trophic functioning of aquatic ecosystems is modelled as a continuous flow of biomass surging up the foodweb, from lower to higher TLs, through predation and ontogenic processes. All the equations of the model are detailed in Gascuel et al. (2011). Such an approach, wherein species as such disappear, may be viewed as the final stage in the use of the TL metric for ecosystem modelling. It provides a simplified but useful representation of ecosystem functioning and impact of fishing. Thus, ET has been used both in theoretical contexts based on virtual ecosystems (Gascuel and Pauly, 2009; Gascuel et al., 2011), or in specific case studies to assess the current fishing impacts at the ecosystem scale (Gasche et al., 2012, in the South African Benguela ecosystem, Lassalle et al., 2012, in the Bay of Biscay, Tremblay-Boyer et al., 2011, for a worldwide analysis), or to analyze the effects of marine protected areas on the whole food web (Colléter et al., 2012 in Sénégal, Valls et al., 2012 in the Mediterranean Sea). Furthermore, ET enables the construction of a unique comparison framework for Ecopath models, the trophic spectrum. This display, based on ET key ideas, is a graphical representation of the ecosystem parameters, such as biomass, production, catch, fishing mortality, etc., along the trophic level (Gascuel et al., 2005). Examples of the use of the ET model and the associated trophic spectra analyses are provided in Figure 1.

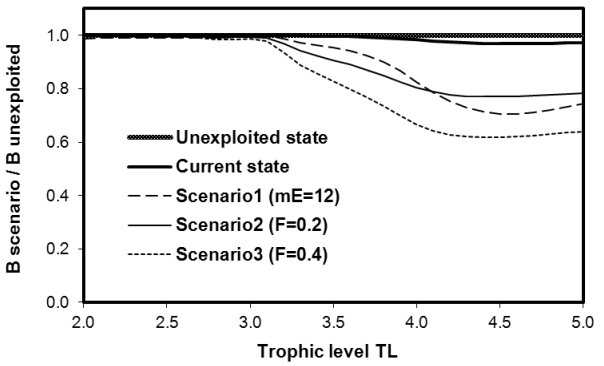
This package is the first attempt to offer interfacing between R and Ecopath through its plug-in



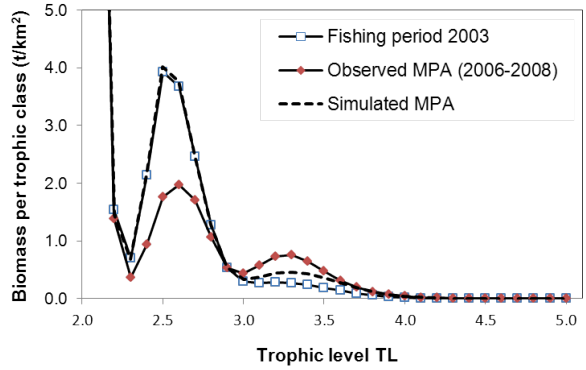
(a) Fishing impact on biomass in the Guinean ecosystem. Simulations refer to various multipliers (mE) of the current (i.e. 2004) fishing mortalities, from mE = 0 (i.e. virgin state) to mE= 5. Results highlight the strong impact of the current fishing effort (mE = 1, dashed lines) on the high TLs biomass: biomass is divided by 2 compared to the virgin state for all trophic levels higher than 3.8. In contrast, TLs around 3 benefit from a release of predation which approximately counterbalances the fishing impact.



(b) Diagnosis on catch made in the Guinean ecosystem. Relative values of catch simulated for some trophic classes are expressed as a function of multipliers of the current fishing mortalities. Results show that high TLs (TL = 4 and 4.5) are fully exploited as an increase in the fishing pressure leads to a decrease in catch. In contrast, yields could be increased for the lower TLs. In other words, increasing fishing pressure would lead to a decrease in the mean trophic level of catch.



(c) Diagnosis on the effect of restricting fishery in the Port-Cros Marine Protected Area (France, Mediterranean Sea): trophic spectra of the relative biomass for a total closure of the fishery and 3 other hypothetical fishing scenarios. Results show that the MPA fulfills its conservation objective with current biomass very close to the virgin state for all trophic levels. Conversely, the other fishing scenarios lead to a significant decrease in the biomass of high trophic levels. It was concluded that the MPA mostly benefits predators and the functional trophic biodiversity.



(d) Analysis of the Marine Protected Area effect in the Bolong de Bamboung (Sénégal). Using the ET-Transpose routine, biomass trophic spectra are built from the 2003 Ecopath model (before the closure of the fishery) and the 2006–2008 model (enforced closure). The ET-Diagnosis routine is used to simulate the MPA effect, starting from the 2003 model and applying a null fishing mortality. Results show that the closure of the fishery can explain, at least partially, the increase observed in the biomass of high TLs ($\geq$ 3.3). The decrease observed for intermediate TLs may not only results from a release in predation. Behavioral (refuge, flee) and environmental effects, that are not included in the model, may thus explain the observed differences.

Figure 1: Examples of the use of the EcoTroph model: diagnosis on the fishing impact on biomass (a) and catches (b) in the Guinean Ecosystem (from Gascuel et al, 2011); assessment of a Marine Protected Area (MPA) in Port-Cros, France (c) (from Valls et al, 2012) and in the Bolong of Bamboung, Sénégal (d) (from Colléter et al., 2012).

EcoTroph. The plug-in will use R and the **EcoTroph** (Guitton et al., 2013) package within the free Ecopath with Ecosim software. This enables the use of ET for a large panel through the plug-in for inexperienced R users, or directly the R package for the more expert ones. On the developer side, this way to link a rich user interface (developed in Microsoft VB) and a well known software in the fishery scientists group such as R is a way to set up a community. These researchers can so focus on the model improvements without taking into account the rich user interface which is time consuming. Along with several functions, we include an example dataset on the Guinean marine ecosystem (Gascuel et al., 2011) within the package. This article introduces, using the example dataset, the three main components of **EcoTroph**:

1. The data import and validation
2. The ET-Transpose tool
3. The ET-Diagnosis tool

A great deal of documentation, both introductory and advanced, is available on the ET website (`http://sirs.agrocampus-ouest.fr/EcoTroph`). The **EcoTroph** package requires the **XML** package (Lang, 2012) in order to load the model input parameters coming from the Ecopath software. The `read.ecopath.model` function will parse the data.

Data import and validation

The **EcoTroph** package requires an input data table to run. The user has to load this dataset under the different possible formats ('.xls', '.csv', '.txt') with the functions `read.table`, `read.csv`... The `ecopath_guinee` dataset (Table 1) is an example of a suitable input table constructed for use with the **EcoTroph** package.

The variable names have to be specified and strictly the same as above: `group_name` (name of the group representing one or several species gathered together), `TL` (the trophic level of the group), `biomass` (the biomass of the group), `prod` (the production on biomass ratio or P/B) and `accessibility` (the ratio of the group biomass that would be caught assuming an infinite fishing pressure). These parameters generally come from Ecopath inputs or outputs, but can also be independent. The entry `catch.1`, `catch.2`, `catch.whatyouwant` is necessary if several fisheries do exist. The `OI` column (the omnivory index, an Ecopath output parameter) is optional, it is used in the `create.smooth` function for an alternative smooth form ($sigmaLN = \frac{OI}{TL_j}$, see below).

The `check.table` function was developed to check the compatibility of the input data table with the **EcoTroph** package:

```
check.table(ecopath_guinee)
```

In the example, no warning message appears as no error is made. If a message appears, the user has to correct the dataset in view of the comments. No missing values (`NA`s) are accepted as input, a yield column has to be entered with 0-values if no catches are registered. As well as the yield, the P/B of detritus groups (not entered in Ecopath) has to be set to 0. The `check.table` function converts the input dataset into a `"data.frame"` object if it is not already one.

A `read.ecopath.model` function was also implemented. It allows users to import data in '.xml' format exported from the Ecopath/EcoTroph plug-in or distributed by a web service (a database of Ecopath models has been set up to allow a meta-analysis at a world scale, `http://sirs.agrocampus-ouest.fr/EcoTroph`). This function formats the inputs so they meet the **EcoTroph** package requirements. Nevertheless, it is advisable to check the data consistency using the `check.table` function.

ET-Transpose tool

Creation of the Smooth function

The Smooth function returns a table allowing the conversion of data referring to specific taxons or functional groups (Ecopath trophic groups for example) into data referring to trophic classes. The major assumption of this function is that distributions of the trophic groups' biomass, yield, etc., around their mean trophic level follow a lognormal curve using the equation:

$$P_{ij} = \frac{1}{(\mathrm{TL}_i - \mathrm{shift})\sigma_j\sqrt{2\pi}} \exp\left[-\frac{(\ln(\mathrm{TL}_i - \mathrm{shift}) - \ln(\overline{\mathrm{TL}_j} - \mathrm{shift}))^2}{2\sigma_j^2}\right]$$

	group_name	TL	biomass	prod	catch.1	catch.2	accessibility	OI
1	Whales	4.01	0.0309	0.020	0.000	0.000	0.0	0.059
2	Dolphins	4.48	0.0433	0.070	0.000	0.000	0.0	0.331
3	Turtles	2.19	0.0296	0.150	0.000	0.000	0.0	0.338
4	Sea birds	3.81	0.0013	0.300	0.000	0.000	0.0	0.353
5	Rays+	3.97	0.3860	0.363	0.012	0.024	0.9	0.329
6	Sharks+	4.31	0.1050	0.410	0.007	0.003	0.8	0.633
7	Large pelagics	4.21	0.3840	0.850	0.025	0.069	0.8	0.263
8	Barracudas+	4.12	0.0583	0.920	0.009	0.022	0.9	0.259
9	Carangids	4.16	0.0627	1.000	0.010	0.024	0.8	0.139
10	Horse Makerels+	3.13	2.3330	0.700	0.000	0.115	0.8	0.366
	...	...	...	...	...	...	...	...
34	Primary producers	1.00	69.0000	84.000	0.000	0.000	0.0	0.000
35	Detritus	1.00	290.0000	0.000	0.000	0.000	0.0	0.193

Table 1: Data from the Ecopath model of the Guinean ecosystem (extracts).

The lognormal distribution is defined by: a mean (the mean trophic level of the group, $\overline{\mathrm{TL}}_j$), a standard deviation (σ_j denoted as `sigmaLN` in the R code) which is a measure of the trophic level variability within the group, and a shift parameter defining the theoretical trophic level characterised by a null variability in TL within group.

The `create.smooth` function enables the creation of this Smooth function using several input parameters. The parameter `ecopath` corresponds to the input data table (`ecopath_guinee` in the example). The parameter `pas` defining the splitting of trophic classes has by default a value of 0.1. The parameter `smooth_type` defines the form of the standard deviation (`sigmaLN`) wanted for the lognormal distribution. Three options are implemented:

1. If `smooth_type=1` (choice by default), `sigmaLN` is constant. This constant `sigmaLN` is equal to the parameter `sigmaLN_cst` specified in the function, and has by default a value of 0.12. The `shift` parameter is set equal to 1.8 by default.
2. If `smooth_type=2`, this is equivalent to `sigmaLN=smooth_param*ln(TL-0.05)`. The parameter `smooth_param` of this formula (also a parameter of the `create.smooth` function) defines the slope of the log-linear trophic level variability increase around the mean trophic level of the group. Based on our experience gained partially through observations, default parameters have been defined as follows: `smooth_param=0.07` and `shift=0.95`. (No need to change the `shift` value in the function, let `shift=NULL`, it will be automatically set to 0.95 for `smooth_type=2`. Same thing for the parameter `smooth_param`.)
3. If `smooth_type=3`, `sigmaLN` for each group is equal to the omnivory index calculated by Ecopath divided by the mean trophic level of the group. A warning message will appear if OIs are equal to 0, they will be automatically changed to a value of 0.01. The parameter `shift` is by default equal to 0. (No need to change the `shift` value in the function, let `shift=NULL`, it will be automatically set to 0 for `smooth_type=3`.)

The `create.smooth` function returns a table of the distribution of each mean trophic level within trophic classes (i.e. how a given species or ecological group, characterized by a given and known mean trophic level, is distributed around this trophic level). This table will be used in the next step of the analysis for the construction of trophic spectra.

```
# default choice, constant sigmaLN
create.smooth(ecopath_guinee)
# sigmaLN = smooth_param*ln(TL-0.05)
create.smooth(ecopath_guinee, smooth_type = 2)
```

A graphic function, `plot(smooth)`, was developed in order to display this Smooth function. The input parameter is the table returned by the `create.smooth` function. It returns a plot with the lognormal curve for each present trophic class (see Figure 2).

```
plot(create.smooth(ecopath_guinee))
```

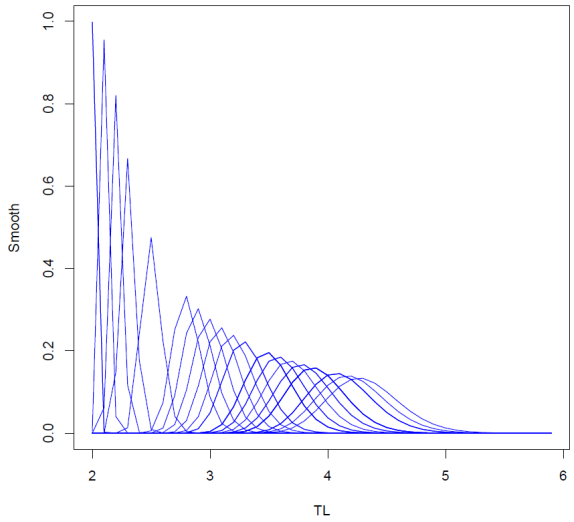


Figure 2: Output of the `plot(smooth)` graphic function, applied to the Guinean ecosystem example. Each curve represents the distribution across trophic levels for all groups whose mean trophic level is equal to the mean value of the distribution. Only curves related to existing groups are displayed.

Data transposition

The `Transpose` function enables the conversion of data referring to specific taxons or functional groups (Ecopath trophic groups for example) into data referring to trophic classes. This function uses the table returned by the `create.smooth` function. The concerned variables are the biomasses, or the catches or others ... Using the `Transpose` function, these variables are distributed continuously along the trophic classes for each group. This function will be reused in the `create.ETmain` function to build a summary table with all the variables calculated by trophic class.

`Transpose` takes as input parameters the table returned by the `create.smooth` function (`tab_smooth`), the input data table (`ecopath`), and the name of the column the user wants to distribute by trophic class (`column`):

```
A <- create.smooth(ecopath_guinee)
# Transpose of the biomass column
T_biomass <- Transpose(A, ecopath_guinee, "biomass")
# Transpose of the catch.1 column
Transpose(A, ecopath_guinee, "catch.1")
```

Results can be displayed graphically using the `plot(Transpose)` function. It takes as input parameter the table returned by the `Transpose` function (`tab_Trans`). The user has the possibility to use a log scale for the y-axis (`scale=log`, the minimum value considered on the graph is conventionally set up at 1/10000 of the total biomass), and to enter a title (`title`):

`plot(tab_Trans,title = NULL,scale = NULL)` returns the principal plots according to the selected column, in particular a plot by group and the associated trophic spectra (see Figure 3).

```
# title and log scale for the biomass
plot(T_biomass, title = "biomass", log)
```

ET_Main table creation

The `create.ETmain` function enables the creation of a summary table, `ET_Main`, containing the principal variables by trophic class. This function can be used directly, there is no need of the previous steps Smooth and Transpose (however these steps are necessary to a good understanding of the EcoTroph model). It takes as input parameters the input data table (`ecopath`) and the parameters of the `create.smooth` function:

```
# constant sigmaLN
create.ETmain(ecopath_guinee)
# sigmaLN = smooth_param*ln(TL-0.05)
create.ETmain(ecopath_guinee, smooth_type = 2)
```

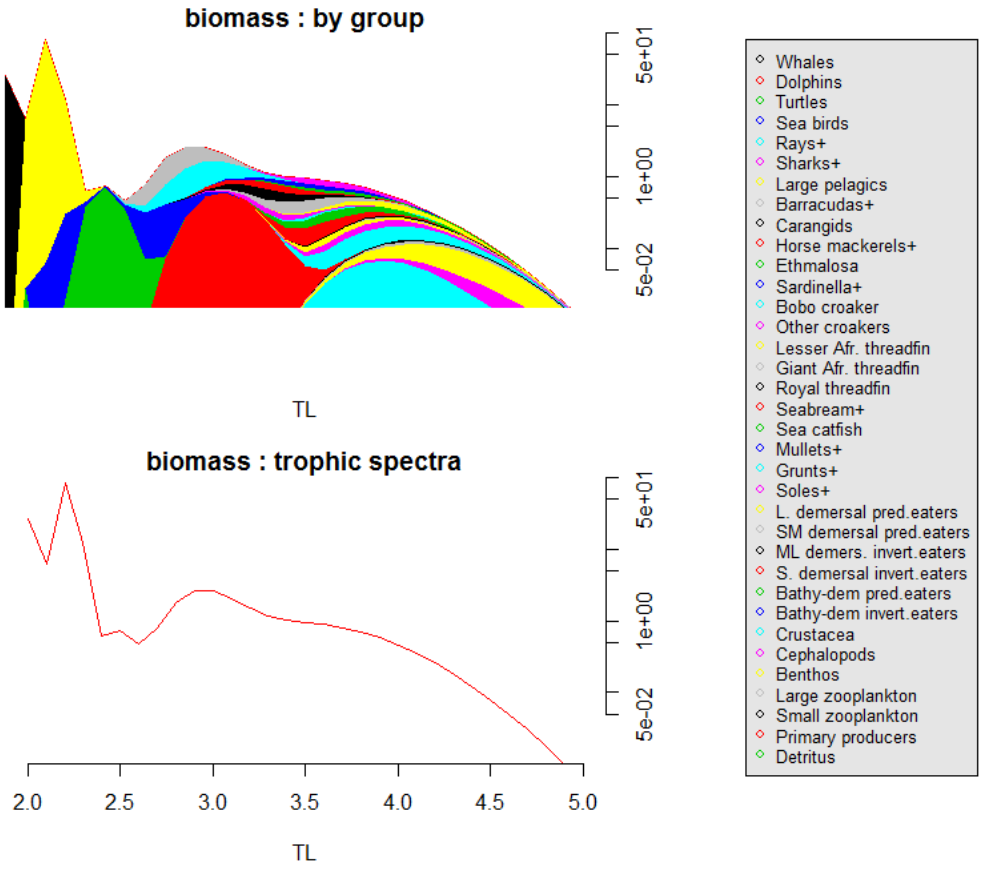


Figure 3: Output of the `plot(Transpose)` graphical function applied to the biomass data per ecological group in the Guinean ecosystem (`ecopath_guinee` dataset). The top panel displays the distributions for each group, while the bottom panel displays the biomass trophic spectrum (BTS, i.e. the sum of all groups).

TL	B	B_acc	P	P_acc	Kin	Kin_acc	Y_tot	F_loss
1	359.00	0.00	5796.00	0.00	16.14	1.00	0.00	0.00
2	25.73	0.00	1029.20	0.00	40.00	0.68	0.00	0.00
2.1	6.26	0.00	13.40	0.00	2.14	0.68	0.00	0.00
2.2	82.06	0.06	147.64	0.04	1.80	0.68	0.01	0.00
2.3	11.94	0.27	21.17	0.20	1.77	0.74	0.05	0.00
2.4	0.62	0.44	0.90	0.57	1.44	1.31	0.24	0.00
	...	...	...	...	...	...	...	...

TL	F_loss_acc	N_loss	Fish_mort	Fish_mort_acc	Selec	Time	N_loss_acc
1	0.00	1.73	0.00	0.00	0.00	0.00	NaN
2	0.22	43.41	0.00	0.15	0.00	0.06	−197.08
2.1	0.22	−24.00	0.00	0.15	0.00	0.06	−70.66
2.2	0.22	19.42	0.00	0.15	0.00	0.11	−16.80
2.3	0.25	31.57	0.00	0.18	0.02	0.17	−10.82
2.4	0.42	−1.89	0.38	0.55	0.70	0.22	−6.45
	...	...	...	...	...	...	...

Table 2: `create.ETmain(ecopath_guinee)$ET_Main` results (extracts).

In the `ET_Main` output table (Table 2), some parameters directly come from the input table (biomass `B`, catch `Y_tot`), while some are calculated based on the definitions of EcoTroph parameters (e.g. Fishing mortality $Fish\_mort = Y/B$, fishing loss rate $F\_loss = Y/P$, see details in Gascuel et al., 2009).

```
# constant sigmaLN
ET_Main <- create.ETmain(ecopath_guinee)\$ET_Main
ET_Main
```

The `create.ETmain` function also returns intermediate tables, i.e. the following tables are also contained in the returned list object:

- `biomass`, the table returned by the `Transpose` function for the column biomass.
- `biomass_acc`, the table `biomass` multiplied by the `accessibility` parameter.
- `prod`, the table corresponding to the production by trophic class.
- `prod_acc`, the table `prod` multiplied by the `accessibility` parameter.
- `tab_smooth`, the table returned by the `create.smooth` function.
- `Y`, the table(s) returned by the `Transpose` function for the column `catch.(x)`.

```
> names(create.ETmain(ecopath_guinee))
[1] "ET_Main" "biomass" "biomass_acc" "prod" "prod_acc" "tab_smooth" "Y"
```

As previously, we developed a graphic function to display the main results: `plot(ETmain)`. Different plots are created, e.g. the biomass trophic spectrum (BTS), the accessible biomass trophic spectrum (ABTS), the catch trophic spectrum (CTS). It takes as input parameter the list object returned by the `create.ETmain` function.

```
plot(create.ETmain(ecopath_guinee),log)
```

Naturally all the returned graphics are not exhaustive. The user can construct other ones using the returned list object. Moreover the `plot(ETmain)` function is implemented with a log scale parameter for the different trophic spectra. However, this does not always provide a good representation. We really encourage users to test different scales for the y-axis. Some plots could be falsely interpreted with no awareness of the y-axis form importance.

ET-Diagnosis simulation tool

ET-Diagnosis is used to simulate the effect of different fishery mortality scenarios on trophic spectra coming from ET-Transpose (see above). Thus different effort multipliers (`Mul_eff` variable), ranging conventionally from 0 to 5, are applied to the initial fishing mortalities $F\tau$. `Mul_eff=0` corresponds

to a fishery closure (F = 0), and allows users to rebuilt an estimate of the unexploited status of the studied ecosystem. Flow equations enable the calculation of the biomasses $B\tau$, the productions $P\tau$ and the catches $Y\tau$ at the equilibrium for each trophic class and `Mul_eff`. The other variables contained in the `ET_Main` table are also treated. We so obtain all the trophic spectra representing the situation at equilibrium for each `Mul_eff`. Effects of fishery mortality changes at an ecosystem scale include biomass, accessible biomass, and kinetic changes but also impacts on the mean trophic level of the catch and the total biomass. The model provides an overview of the current fishing impact on the ecosystem (compared to the unexploited state), and some long term forecasts on the consequences of increasing or decreasing fishing pressures. This model also enables to see how different ecosystem functioning hypotheses (values of the extent recycling, top-down effect) could affect the ecosystem and trophic scale properties.

ET-Diagnosis is implemented through the `create.ETdiagnosis` function. It takes as input parameters the list object returned by the `create.ETmain` function, `Mul_eff` a vector of the different effort multipliers, and the specific parameters of the ET-diagnosis simulations:

- `Beta`, a parameter taking values between 0 and 1, which defines the intensity of the biomass recycling by the microbial loop (default value set to 0.1)
- `TopD`, a parameter taking values between 0 and 1, which defines the intensity of the top-down control of predators on their preys (default value set to 0.2)
- `FormD`, a parameter taking values between 0 and 1, which defines the functional relationship between preys and predators (default value set to 0.5)

(cf. package help for more details).

This function returns two types of results for each simulated effort multiplier: indices calculated at the ecosystem scale for each effort multiplier, and the catches, biomasses, accessible biomasses, productions . . . for each trophic class.

```
ETmain <- create.ETmain(ecopath_guinee)
create.ETdiagnosis(ETmain)
# change of the top-down parameter
create.ETdiagnosis(ETmain, TopD = 0.6)
```

A graphic function, `plot(m,scale=null,maxrange=null)`, displays the principal plots coming from the function `create.ETdiagnosis`: the biomass, predator biomass, catches ratio plots for the different effort multipliers, and the principle trophic spectra. This function takes as input parameters the list object returned by the `create.ETdiagnosis` function (`m`), one scale parameter for the y-axis of the BTS (`scale`, log or not), and the maximum wanted for the x-axis (`maxrange`). Naturally all the returned graphics are not exhaustive as users can construct other ones using the returned list object. As previously mentioned, we really encourage users to test different scales for the y-axis. Figure 4 is one major graphic of the eight displayed by the function.

```
# log scale for the BTS
diag <- create.ETdiagnosis(ETmain)
plot(diag, log)
```

Summary

This article describes the principle functions of the **EcoTroph** package. The package enables the analysis of fishing impacts on aquatic trophic networks in a simple way, and also the simulation of various fisheries in terms of catch (see Figure 1). New simulation tools are currently being developed to improve the model and enable the implementation of more options regarding the fisheries scenarios. The package is available on the Comprehensive R Archive Network (CRAN, http://CRAN.R-project.org/) and users are encouraged to provide feedback in order to enhance the tool. All the information contained in this article is not exhaustive. More details are available in the package help, and on the website (http://sirs.agrocampus-ouest.fr/EcoTroph). Feel free to contact the authors for any problem encountered while using the package.

Acknowledgement

We would like to thank all the persons involved in the development of EcoTroph. There are now numerous applications of this model available in the scientific literature and we hope the R package will help to continue to forge ahead.

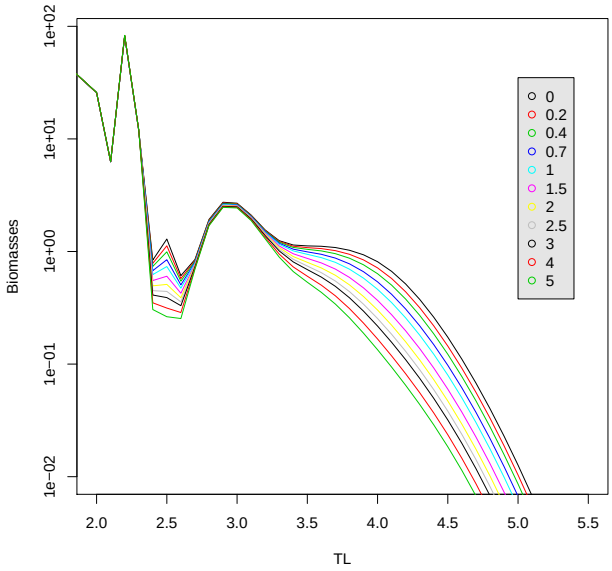


Figure 4: Simulated biomass trophic spectra (BTS) of the Guinean ecosystem for various fishing effort multipliers ranging from 0 (no fishing) to 5. Such a graph highlights the fishing effects on trophic levels around 2.5 or higher than 3.4, with larger impact for the top predators (highest TLs). In contrast, the low and intermediate ones are less damaged.

Bibliography

V. Christensen and D. Pauly. ECOPATH II—a software for balancing steady-state ecosystem models and calculating network characteristics. *Ecological Modelling*, 61(3–4):169–185, 1992. [p98]

M. Colléter, D. Gascuel, J. Ecoutin, and L. Tito de Morais. Modelling trophic flows in ecosystems to assess the efficiency of marine protected area (MPA), a case study on the coast of Sénégal. *Ecological Modelling*, 232:1–13, 2012. [p98]

C. Elton. *Animal Ecology*. The Macmillan Company, New York, 1927. [p98]

L. Gasche, D. Gascuel, L. Shannon, and Y. Shin. Global assessment of the fishing impacts on the Southern Benguela ecosystem using an EcoTroph modelling approach. *Journal of Marine Systems*, 90 (1):1–12, 2012. [p98]

D. Gascuel. The trophic-level based model: A theoretical approach of fishing effects on marine ecosystems. *Ecological Modelling*, 189(3–4):315–332, 2005. [p98]

D. Gascuel and D. Pauly. EcoTroph: Modelling marine ecosystem functioning and impact of fishing. *Ecological Modelling*, 220(21):2885–2898, 2009. [p98]

D. Gascuel, Y.-M. Bozec, E. Chassot, A. Colomb, and M. Laurans. The trophic spectrum: Theory and application as an ecosystem indicator. *ICES Journal of Marine Science*, 62(3):443–452, 2005. [p98]

D. Gascuel, L. Tremblay-Boyer, and D. Pauly. EcoTroph (ET): A trophic level based software for assessing the impacts of fishing on aquatic ecosystems. *Fisheries Centre Research Reports*, 17(1):1–82, 2009. [p98, 104]

D. Gascuel, S. Guénette, and D. Pauly. The trophic-level-based ecosystem modelling approach: Theoretical overview and practical uses. *ICES Journal of Marine Science*, 68(7):1403–1416, 2011. [p98, 100]

J. Guitton, M. Colléter, D. Gascuel, and P. Gatti. *EcoTroph: EcoTroph modelling support*, 2013. URL http://sirs.agrocampus-ouest.fr/EcoTroph/. R package version 1.5-1. [p100]

D. T. Lang. *XML: Tools for parsing and generating XML within R and S-Plus.*, 2012. URL http://CRAN.R-project.org/package=XML. R package version 3.9-4. [p100]

G. Lassalle, D. Gascuel, F. Le Loc'h, J. Lobry, G. J. Pierce, V. Ridoux, M. B. Santos, J. Spitz, and N. Niquil. An ecosystem approach for the assessment of fisheries impacts on marine top predators: The Bay of Biscay case study. *ICES Journal of Marine Science*, 69(6):925–938, 2012. [p98]

R. L. Lindeman. The trophic-dynamic aspect of ecology. *Ecology*, 23(4):399–417, 1942. [p98]

W. E. Odum and E. J. Heald. The detritus-based food web of an estuarine mangrove community. In L. E. Cronin, editor, *Estuarine Research*, volume 1, pages 265–286. Academic Press, New York, 1975. [p98]

J. J. Polovina. An overview of the ECOPATH model. *Fishbyte*, 2(2):5–7, 1984. [p98]

L. Tremblay-Boyer, D. Gascuel, R. Watson, V. Christensen, and D. Pauly. Modelling the effects of fishing on the biomass of the world's oceans from 1950 to 2006. *Marine Ecology Progress Series*, 442: 169–185, 2011. [p98]

A. Valls, D. Gascuel, S. Guénette, and P. Francour. Modeling trophic interactions to assess the effects of a marine protected area: Case study in the NW Mediterranean Sea. *Marine Ecology Progress Series*, 456:201–214, 2012. [p98]

Mathieu Colléter
Université Européenne de Bretagne, Agrocampus Ouest, UMR985 ESE, INRA, Agrocampus Ouest
Rennes
France
and
The Fisheries Center, University of British Colombia, Vancouver
Canada
mathieu.colleter@agrocampus-ouest.fr

Jérôme Guitton
Université Européenne de Bretagne, Agrocampus Ouest, UMR985 ESE, INRA, Agrocampus Ouest
Rennes
France
jerome.guitton@agrocampus-ouest.fr

Didier Gascuel
Université Européenne de Bretagne, Agrocampus Ouest, UMR985 ESE, INRA, Agrocampus Ouest
Rennes
France
didier.gascuel@agrocampus-ouest.fr

stellaR: A Package to Manage Stellar Evolution Tracks and Isochrones

by Matteo Dell'Omodarme and Giada Valle

Abstract We present the R package **stellaR**, which is designed to access and manipulate publicly available stellar evolutionary tracks and isochrones from the Pisa low-mass database. The procedures for extracting important stages in the evolution of a star from the database, for constructing isochrones from stellar tracks and for interpolating among tracks are discussed and demonstrated.

Due to the advance in the instrumentation, nowadays astronomers can deal with a huge amount of high-quality observational data. In the last decade impressive improvements of spectroscopic and photometric observational capabilities made available data which stimulated the research in the globular clusters field. The theoretical effort of recovering the evolutionary history of the clusters benefits from the computation of extensive databases of stellar tracks and isochrones, such as Pietrinferni et al. (2006); Dotter et al. (2008); Bertelli et al. (2008). We recently computed a large data set of stellar tracks and isochrones, "The Pisa low-mass database" (Dell'Omodarme et al., 2012), with up to date physical and chemical inputs, and made available all the calculations to the astrophysical community at the Centre de Données astronomiques de Strasbourg (CDS)[1], a data center dedicated to the collection and worldwide distribution of astronomical data.

In most databases, the management of the information and the extraction of the relevant evolutionary properties from libraries of tracks and/or isochrones is the responsibility of the end users. Due to its extensive capabilities of data manipulation and analysis, however, R is an ideal choice for these tasks. Nevertheless R is not yet well known in astrophysics; up to December 2012 only seven astronomical or astrophysical-oriented packages have been published on CRAN (see the CRAN Task View *Chemometrics and Computational Physics*).

The package **stellaR** (Dell'Omodarme and Valle, 2012) is an effort to make available to the astrophysical community a basic tool set with the following capabilities: retrieve the required calculations from CDS; plot the information in a suitable form; construct by interpolation tracks or isochrones of compositions different to the ones available in the database; construct isochrones for age not included in the database; extract relevant evolutionary points from tracks or isochrones.

Get stellar evolutionary data

The Pisa low-mass database contains computations classified according to four parameters: the metallicity z of the star, its initial helium value y, the value of α-enhancement of the heavy elements mixture with respect to the reference mixture and the mixing-length parameter α_{ml} used to model external convection efficiency. The values of the parameters available in the database can be displayed using the function `showComposition()`:

```
> showComposition()
Mixing-length values:
         1.7, 1.8, 1.9

alpha-enhancement values:
         0, 1  (i.e. [alpha/Fe] = 0.0 [alpha/Fe] = 0.3)

Chemical compositions:
      z    y.1   y.2   y.3   y.4   y.5   y.6
  1e-04  0.249  0.25  0.27  0.33  0.38  0.42
  2e-04  0.249  0.25  0.27  0.33  0.38  0.42
  3e-04  0.249  0.25  0.27  0.33  0.38  0.42
  4e-04  0.249  0.25  0.27  0.33  0.38  0.42
  5e-04  0.250  0.25  0.27  0.33  0.38  0.42
  6e-04  0.250  0.25  0.27  0.33  0.38  0.42
  7e-04  0.250  0.25  0.27  0.33  0.38  0.42
  8e-04  0.250  0.25  0.27  0.33  0.38  0.42
  9e-04  0.250  0.25  0.27  0.33  0.38  0.42
  1e-03  0.250  0.25  0.27  0.33  0.38  0.42
```

[1] via anonymous ftp from ftp://cdsarc.u-strasbg.fr or via http://cdsarc.u-strasbg.fr/viz-bin/qcat?J/A+A/540/A26

```
  2e-03  0.252  0.25  0.27  0.33  0.38  0.42
  3e-03  0.254  0.25  0.27  0.33  0.38  0.42
  4e-03  0.256  0.25  0.27  0.33  0.38  0.42
  5e-03  0.258  0.25  0.27  0.33  0.38  0.42
  6e-03  0.260  0.25  0.27  0.33  0.38  0.42
  7e-03  0.262  0.25  0.27  0.33  0.38  0.42
  8e-03  0.264  0.25  0.27  0.33  0.38  0.42
  9e-03  0.266  0.25  0.27  0.33  0.38  0.42
  1e-02  0.268  0.25  0.27  0.33  0.38  0.42
```

The table of chemical compositions presents all the y values available for a given z. For a set of parameters, the track files are identified specifying the mass of the desired model (in the range [0.30 - 1.10] $M_\odot$ ($M_\odot = 1.99 \cdot 10^{33}$ g is the mass of the Sun), in steps of 0.05 $M_\odot$), while the age (in the range [8.0 - 15.0] Gyr, in steps of 0.5 Gyr) is required for the isochrones.

Upon specification of the aforementioned parameters, the **stellaR** package can import data from CDS (via anonymous ftp) over an active Internet connection. The CDS data are stored in ASCII format and include a header with calculation metadata, such as the metallicity, the initial helium abundance, and the mixing-length. The import is done via a `read.table()` call, skipping the header of the files.

The following data objects can be downloaded from the database site:

- Stellar track: a stellar evolutionary track computed starting from Pre-Main Sequence (PMS) and ending at the onset of helium flash (for masses $M \geq 0.55\ M_\odot$) or at the exhaustion of central hydrogen (for $0.30\ M_\odot \leq M \leq 0.50\ M_\odot$). The functions `getTrk()` and `getTrkSet()` can be used to access such data; they respectively return objects of classes `"trk"` and `"trkset"`.
- Stellar ZAHB: Zero-Age Horizontal-Branch models. The function `getZahb()` can be used to access such data; it returns an object of class `"zahb"`.
- HB models: computed from ZAHB to the onset of thermal pulses. The functions `getHb()` and `getHbgrid()` can be used to access such data; they respectively return objects of classes `"hb"` and `"hbgrid"`.
- Stellar isochrones: computed in the age range [8.0 - 15.0] Gyr. The functions `getIso()` and `getIsoSet()` can be used to access such data; they respectively return objects of classes `"iso"` and `"isoset"`.

Readers interested in details about the computation procedure are referred to Dell'Omodarme et al. (2012). The data gathered from CDS are organized into objects of appropriate classes. The package includes `print` and `plot` S3 methods for the classes `"trk"`, `"trkset"`, `"zahb"`, `"hb"`, `"hbgrid"`, `"iso"`, and `"isoset"`.

As an example, we illustrate the recovering of the stellar track for a model of mass $M = 0.80\ M_\odot$, metallicity $z = 0.001$, initial helium abundance $y = 0.25$, mixing-length $\alpha_{\rm ml} = 1.90$, α-enhancement $[\alpha/{\rm Fe}] = 0.0$.

```
> track <- getTrk(m = 0.80, z = 0.001, y = 0.25, ml = 1.90, afe = 0)
> track
         Stellar track

Mass = 0.8 Msun
Z = 0.001 , Y = 0.25
Mixing length = 1.9
[alpha/Fe] = 0

> names(track)
[1] "mass"      "z"         "y"         "ml"        "alpha.enh" "data"
> class(track)
[1] "trk"     "stellar"
```

The function `getTrk()` returns an object of class `"trk"`, which is a list containing the track metadata, i.e. the star mass, the metallicity, the initial helium abundance, the mixing-length and the α-enhancement, and the computed data in the data frame `data`. Track data contains the values of 15 variables:

```
> names(track$data)
 [1] "mod"     "time"    "logL"    "logTe"   "mass"    "Hc"      "logTc"   "logRHOc"
 [9] "MHEc"    "Lpp"     "LCNO"    "L3a"     "Lg"      "radius"  "logg"
```

The included variables are: `mod` the progressive model number; `time` the logarithm of the stellar age (in yr); `logL` the logarithm of the surface luminosity (in units of solar luminosity); `logTe` the logarithm of

the effective temperature (in K); `mass` the stellar mass (in units of solar mass); `Hc` the central hydrogen abundance (after hydrogen exhaustion: central helium abundance); `logTc` the logarithm of the central temperature (in K); `logRHOc` the logarithm of the central density (in g/cm^3); `MHEc` the mass of the helium core (in units of solar mass); `Lpp` the luminosity of pp chain (in units of surface luminosity); `LCNO` the luminosity of CNO chain (in units of surface luminosity); `L3a` the luminosity of triple-α burning (in units of surface luminosity); `Lg` luminosity of the gravitational energy (in units of surface luminosity); `radius` the stellar radius (in units of solar radius); `logg` the logarithm of surface gravity (in cm/s^2).

Similarly the part of the track starting from ZAHB and ending at the onset of thermal pulses can be downloaded with the call:

```
> hbtk <- getHb(m = 0.80, z = 0.001, y = 0.25, ml = 1.90, afe = 0)
> hbtk
         Stellar track from ZAHB

Mass = 0.8 Msun
Mass RGB = 0.8 Msun
Z = 0.001 , Y = 0.25
Mixing length = 1.9
[alpha/Fe] = 0

> names(hbtk)
[1] "mass"      "massRGB"   "z"         "y"         "ml"        "alpha.enh"
[7] "data"
> class(hbtk)
[1] "hb"      "stellar"
```

Function `getHb()` returns an object of class `"hb"`, which differs from an object of class `"trk"` only for the presence of the variable `massRGB`, i.e. the Red-Giant Branch (RGB) progenitor mass.

Usually a set of tracks with different mass and/or metallicity values are needed for computations. The package **stellaR** provides the function `getTrkSet()`, which can download a set of tracks with different values for mass, metallicity, initial helium abundance, mixing-length and α-enhancement. As an example the whole set of masses (from 0.30 to 1.10 $M_\odot$, in steps of 0.05 $M_\odot$), for metallicity $z = 0.001$, initial helium abundance $y = 0.25$, mixing-length $\alpha_{\rm ml} = 1.90$, and α-enhancement [α/Fe] = 0.0 can be downloaded as follows:

```
> mass <- seq(0.3, 1.1, by = 0.05)
> trks <- getTrkSet(m = mass, z = 0.001, y = 0.25, ml = 1.90, afe = 0)
> trks
[[1]]
         Stellar track

Mass = 0.3 Msun
Z = 0.001 , Y = 0.25
Mixing length = 1.9
[alpha/Fe] = 0

[[2]]
         Stellar track

Mass = 0.35 Msun
Z = 0.001 , Y = 0.25
Mixing length = 1.9
[alpha/Fe] = 0

...
```

The function `getTrkSet()` returns an object of class `"trkset"`, a list containing objects of class `"trk"`. The track set can be displayed in the usual ($\log T_{\rm eff}$, $\log L/L_\odot$) plane by a call of the function `plot()`:

```
> plot(trks, lty = 1:2)
```

The output of the function is shown in Figure 1. The plot is produced by a call to the function `plotAstro()`, which allows the user to customize several aspects of the plot, such as the axes labels, the number of minor ticks between two major ticks, the limits of the axes, the color and type of the lines (as in the example), the type of the plot (lines, points, both, ...).

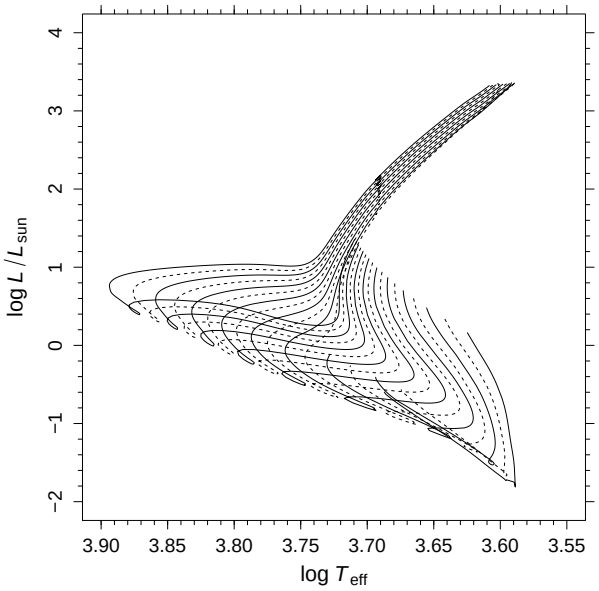


Figure 1: The evolutionary tracks for masses from $M = 0.30\ M_{\odot}$ to $M = 1.10\ M_{\odot}$ from PMS to He flash. The parameters of the calculations are: $z = 0.001$, $y = 0.25$, $\alpha_{\rm ml} = 1.90$, $[\alpha/{\rm Fe}] = 0.0$.

The use of the `plot()` function is further demonstrated in Figure 2, where, for $z = 0.001$, $y = 0.25$, $\alpha_{\rm ml} = 1.90$, $[\alpha/{\rm Fe}] = 0.0$, the evolutionary tracks for $M = 0.80\ M_{\odot}$ from PMS to He flash (black line) and from ZAHB to thermal pulses (green line) are displayed. The figure is obtained as follows:

```
> plot(track)
> plot(hbtk, add = TRUE, col = "green")
```

Apart from the plots discussed before, it is easy to display other relations between the computed variables. In the following example we get the data for two masses, namely 0.50 and 1.00 $M_{\odot}$, and plot the trend of the radius (in units of solar radius) versus the logarithm of the age for the first 100 models. The resulting plot is displayed in Figure 3.

```
> trkr <- getTrkSet(m = c(0.5, 1), z = 0.01, y = 0.25, ml = 1.8, afe = 0)
> mydata <- do.call(rbind, lapply(trkr, "[[", "data"))
> D <- subset(mydata, mod <= 100)
> key <- as.numeric(factor(D$mass))
> plotAstro(D$time, D$radius, type = "p", pch = key, ylab = "Radius (Rsun)",
+    xlab = "log age (yr)")
> legend("topright", c("M=0.50", "M=1.00"), pch = 1:2)
```

Isochrones can be obtained from CDS and plotted in a similar way. As an example, we get isochrones of 9 and 12 Gyr for $z = 0.001$, $y = 0.25$, $\alpha_{\rm ml} = 1.90$, $[\alpha/{\rm Fe}] = 0.0$:

```
> isc <- getIsoSet(age = c(9, 12), z = 0.001, y = 0.25, ml = 1.90, afe = 0)
> isc
[[1]]
         Stellar isochrone

Age = 9 Gyr
Z = 0.001 , Y = 0.25
Mixing length = 1.9
[alpha/Fe] = 0

[[2]]
         Stellar isochrone

Age = 12 Gyr
Z = 0.001 , Y = 0.25
Mixing length = 1.9
[alpha/Fe] = 0
```

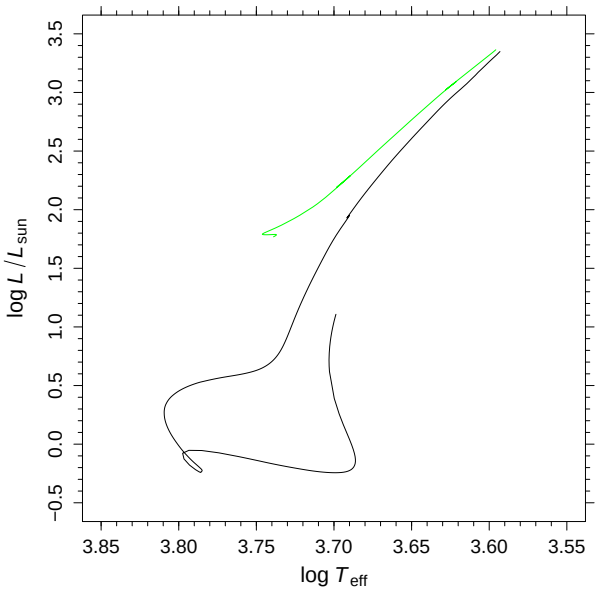


Figure 2: Black line: evolutionary tracks for mass $M = 0.80\ M_{\odot}$, $z = 0.001$, $y = 0.25$, $\alpha_{\mathrm{ml}} = 1.90$, [$\alpha$/Fe] = 0.0 from PMS to He flash. Green line: evolutionary track from ZAHB to thermal pulses.

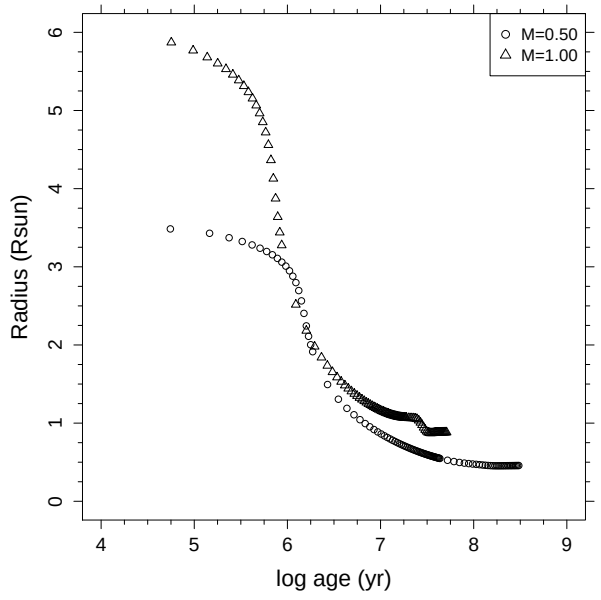


Figure 3: Radius versus the logarithm of the age for the first 100 models of two stars with different mass ($M = 0.50\ M_{\odot}$ and $M = 1.00\ M_{\odot}$) and identical composition, $z = 0.01$, $y = 0.25$, $\alpha_{\mathrm{ml}} = 1.8$, [$\alpha$/Fe] = 0.0.

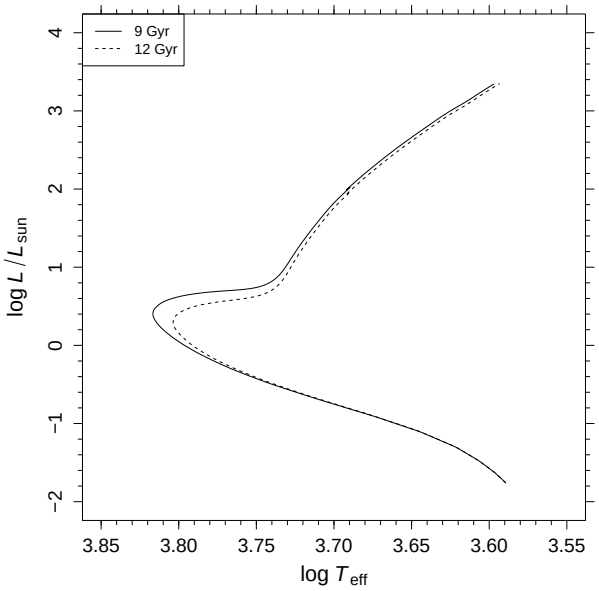


Figure 4: Isochrones in the theoretical plane for 9 and 12 Gyr. Calculations performed for $z = 0.001$, $y = 0.25$, $\alpha_{\rm ml} = 1.90$, $[\alpha/{\rm Fe}] = 0.0$.

```
attr(,"class")
[1] "isoset"  "stellar"

> names(isc[[1]])
[1] "age"       "z"         "y"         "ml"        "alpha.enh" "data"
> names(isc[[1]]$data)
[1] "logL"   "logTe"  "mass"   "radius" "logg"
```

The function returns an object of class `"isoset"`, a list containing objects of class `"iso"`. The latter objects are lists containing metadata (age, metallicity, initial helium abundance, mixing-length, α-enhanchment) and the data frame `data`, which contains the computed theoretical isochrones data. Figure 4 shows the set of isochrones plotted with the commands:

```
> plot(isc, lty = 1:2)
> legend("topleft", c("9 Gyr", "12 Gyr"), lty = 1:2)
```

Tools for interpolating among data structures

Even if the database provides a fine grid of models it is possible that the specific model needed for a comparison with observational data has not been calculated. To address this issue, the package **stellaR** includes tools for constructing new sets of isochrones. The simplest case is whenever one desires an isochrone for ages not included in the database, but for a combination of metallicity, initial helium abundance, mixing-length and α-enhanchment existing in the database. The function `makeIso()` is able to compute the required isochrones by means of interpolation on a set of tracks. The interpolation can be performed for age in the range [7.0 - 15.0] Gyr. The user has the choice to explicitly provide to the function a set of tracks, previously downloaded from CDS, or to specify the required composition of the tracks to be downloaded for the interpolation. To show the usage of the function we use the object `trks` downloaded before to obtain an isochrone of age 9.7 Gyr:

```
> iso.ip <- makeIso(age = 9.7, tr = trks)
> iso.ip
         Stellar isochrone

Age = 9.7 Gyr
Z = 0.001 , Y = 0.25
Mixing length = 1.9
[alpha/Fe] = 0
```

The call produces a result identical to `makeIso(age = 9.7,z = 0.001,y = 0.25,ml = 1.9,afe = 0);` in the latter case the data are taken from CDS before the interpolation procedure.

The interpolation technique is based upon the fact that all the tracks contain the same number of points by construction, and that a given point corresponds to the same evolutionary phase on all the tracks. We define $S(M)$ as the set of tracks to be used for interpolation, parametrized by the value of the mass M. Let $t_i(M)$ be the evolutionary time for the ith point on the track of mass M, and A be the age of the required isochrone. Let k be the point on the track of lower mass of the set $S(M)$ for which $t_k(M) \geq A$. For each point $j \geq k$ on the tracks in $S(M)$, a linear interpolation in age of the values of mass, logarithm of the effective temperature and logarithm of the luminosity is performed among tracks. These points define the required isochrone. A potential problem of this simple procedure will occur whenever massive stars develop a convective core during the Main Sequence (MS). In this case, as shown for example in Mowlavi et al. (2012), the monotonic trend of the evolutionary time – that decreases with increasing stellar mass at the end of the MS – inverts at the middle of the MS. However the problem will be encountered only for early-time isochrones, for which the mass at the isochrone Turn-Off will be in the interval during which the convective core develops. The procedure outlined in this Section is adequate for construction of isochrones throughout the range allowed by the function.

Track interpolation

The package **stellaR** provides also a tool for performing a 3D interpolation on the database to construct a set of tracks for values of metallicity, initial helium abundance and mixing-length not included in the computations available at CDS. The function `interpTrk()` can be used for this procedure. A call to this function causes the download from CDS of the sets of tracks needed for the interpolation.

The new set of tracks is computed by means of a linear interpolation. The metallicity is log-transformed before the interpolation procedure. Let $T_{z,y,\alpha_{\rm ml}}(M)_i$ be the ith point in the data set containing the evolutionary time, the effective temperature and the logarithm of surface luminosity for the track of mass M and given composition. The interpolation algorithm proceeds as follows:

$$\overbrace{T_{z,y,\alpha_{\rm ml}}(M)_i}^{8\text{ sets}} \rightarrow \overbrace{T_{z,y,*}(M)_i}^{4\text{ sets}} \rightarrow \overbrace{T_{z,*,*}(M)_i}^{2\text{ sets}} \rightarrow \overbrace{T_{*,*,*}(M)_i}^{1\text{ set}}$$

The symbol $*$ means that interpolation occurred in the substituted variable. The selection of the set of tracks which enter in the interpolation is based upon the identification of the vertexes of the cell of the (z, y, $\alpha_{\rm ml}$) space containing the point identified by the required parameters. Then, for all the 17 masses at the vertexes, the linear interpolation described above is performed. In the worst case scenario, whenever none of the supplied parameters values exists in the database, the interpolation requires $2^3 = 8$ sets of 17 masses. The algorithm is however able to reduce the dimensionality of the process if some of the variable values exist in the database.

As a demonstration, let us compute a set of tracks with mixing-length value $\alpha_{\rm ml}$ = 1.74, z = 0.002, y = 0.25, [α/Fe] = 0.0:

```
> ip.trk <- interpTrk(z = 0.002, y = 0.25, ml = 1.74, afe = 0)
```

Since the values of z and y exist in the database, only an interpolation on the mixing-length value is performed by the function. The set of tracks can be used for isochrone construction, like a standard set of tracks:

```
> ip.iso <- makeIso(age = 12, tr = ip.trk)
```

Keypoints extraction

Important stages in the evolution of a star are defined as "keypoints", e.g. hydrogen core exhaustion, Turn-Off luminosity, RGB tip luminosity. To simplify their extraction the package **stellaR** provides the function `keypoints()`, which operates on an object of class `"trk"` or `"iso"`.

The function extracts from the data stored in objects of class `"trk"` the rows of the data frame relative to the following evolutionary stages:

1. ZAMS. Zero-Age Main-Sequence, defined as the point for which the central H abundance drops below 99% of its initial value.
2. TO. Turn-Off, defined as the point for which the effective temperature reaches its maximum value. If multiple lines satisfy the constraint, the values of all the rows are averaged.
3. BTO. Brighter Turn-Off, defined as the point for which the effective temperature drops below the temperature of the TO minus 100 K. This point can not exist for low masses. Details on the advantages of this evolutionary point with respect to the TO can be found in Chaboyer et al. (1996).

4. exHc. Central H exhaustion, defined as the point for which the central H abundance is zero. For low masses the point can coincide with TO. This is the last point of the tracks with mass lower or equal to 0.50 $M_{\odot}$.
5. Heflash. Helium flash, the last point of the track for masses higher than 0.50 $M_{\odot}$.

When the function is called on an object of class `"iso"` it returns a data frame containing only TO and BTO phases.

In both cases the function inserts in the returned data frame the columns relative to mass (or age for isochrones), metallicity, initial helium abundance value, mixing-length, α-enhancement, and evolutionary phase identifier.

As a demonstration we extract the TO and BTO points from the object `isc` generated in a previous example:

```
> kp <- keypoints(isc)
> kp
          logL    logTe      mass  radius     logg age     z    y  ml alpha.enh id
TO   0.4044723 3.816652 0.8396903 1.23558 4.178911   9 0.001 0.25 1.9         0  1
BTO  0.5556971 3.809943 0.8561162 1.51671 4.009265   9 0.001 0.25 1.9         0  2
TO1  0.2917250 3.803611 0.7772973 1.15234 4.205965  12 0.001 0.25 1.9         0  1
BTO1 0.4495597 3.796545 0.7909500 1.42768 4.027426  12 0.001 0.25 1.9         0  2
```

The points can be easily superimposed to the isochrones in the theoretical plane. The top panel of Figure 5 is obtained with the following commands:

```
> plot(isc)
> points(logL ~ logTe, data = kp, pch = id, col = id)
> legend("topleft", c("TO", "BTO"), pch = 1:2, col = 1:2)
```

As a last example we extract a set of tracks for masses in the range [0.40 - 1.10] $M_{\odot}$ and three metallicity values z = 0.0001, 0.001, 0.01 and we display the time of exhaustion of central hydrogen as a function of the mass of the star (bottom panel in Figure 5).

```
> mT <- seq(0.4, 1.1, by = 0.1)
> zT <- c(0.0001, 0.001, 0.01)
> tr <- getTrkSet(m = mT, z = zT, y = 0.25, ml = 1.9, afe = 1)
> kp <- keypoints(tr)
> kpH <- subset(kp, id == 4)
> symbol <- as.numeric(factor(kpH$z))
> plotAstro(kpH$M, kpH$time, type = "p", pch = symbol, xi = 0.1,
+    xlab = expression(italic(M) ~ (italic(M)[sun])), ylab = "log age (yr)")
> lab <- format(sort(unique(kpH$z)), nsmall = 4, scientific = FALSE)
> legend("topright", lab, pch = 1:3)
```

Summary

This paper demonstrated how the package **stellaR** can be useful to the astrophysical community as a tool to simplify the access to stellar tracks and isochrone calculations available on-line. A set of tools to access, manipulate and plot data are included in the package and their usage was shown. The interpolation functions included in the package can be used to safely produce tracks or isochrones at compositions not included in the database, without the need that users develop software on their own. A planned extension of the package is the modification of the algorithm of isochrone construction to make the calculation of isochrones of young ages feasible. This step can be useful in view of a possible extension of the Pisa database to higher masses, or to manipulation of data stored in other databases. In fact, while the package is currently developed for accessing data from the Pisa low-mass database, other public databases can be in principle accessed in the same way. This step is however complicated by the fact that no standard for the stellar model output exists in the astrophysical community, requiring the individual adaptation of the interface functions for each data set.

Acknowledgments

We are grateful to our anonymous referees for many stimulating suggestions that helped to clarify and improve the paper and the package. We thank Steve Shore for a careful reading of the manuscript and many useful suggestions.

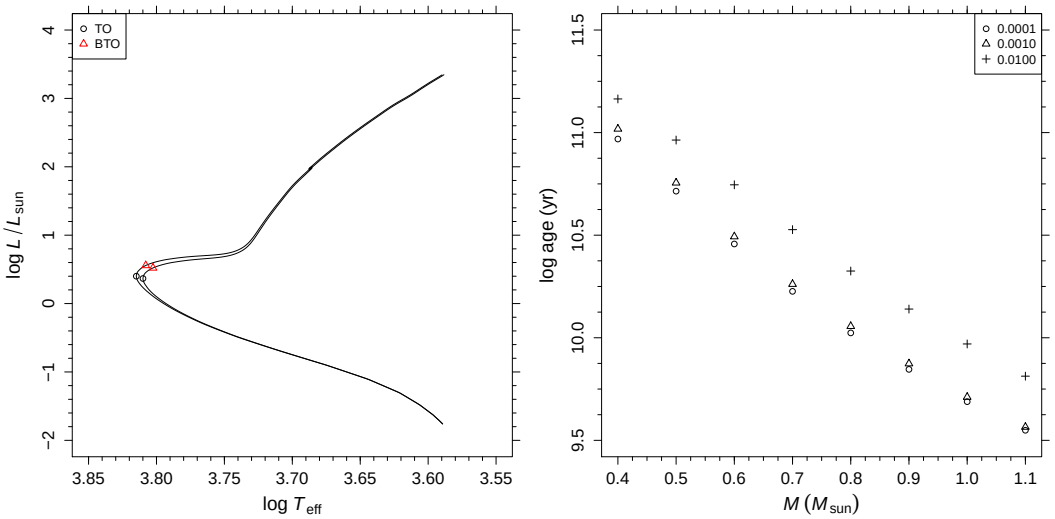


Figure 5: Top panel: same as Figure 4. The position of Turn-Off (TO) and Brighter TO (BTO) are shown. Bottom panel: time to exhaustion of central hydrogen as a function of the mass of the star. The symbols identify the values of the metallicity z from 0.0001 to 0.01.

Bibliography

G. Bertelli, L. Girardi, P. Marigo, and E. Nasi. Scaled solar tracks and isochrones in a large region of the Z–Y plane. I. From the ZAMS to the TP-AGB end for 0.15–2.5 $M_{\odot}$ stars. *Astronomy & Astrophysics*, 484(3):815–830, 2008. [p108]

B. Chaboyer, P. Demarque, P. J. Kernan, L. M. Krauss, and A. Sarajedini. An accurate relative age estimator for globular clusters. *Monthly Notices of the Royal Astronomical Society*, 283(2):683–689, 1996. [p114]

M. Dell'Omodarme and G. Valle. *stellaR: Stellar Evolution Tracks and Isochrones*, 2012. URL http://CRAN.R-project.org/package=stellaR. R package version 0.3-1. [p108]

M. Dell'Omodarme, G. Valle, S. Degl'Innocenti, and P. G. Prada Moroni. The Pisa Stellar Evolution Data Base for low-mass stars. *Astronomy & Astrophysics*, 540:A26, 2012. [p108, 109]

A. Dotter, B. Chaboyer, D. Jevremović, V. Kostov, E. Baron, and J. W. Ferguson. The Dartmouth Stellar Evolution Database. *Astrophysical Journal Supplement Series*, 178(1):89–101, 2008. [p108]

N. Mowlavi, P. Eggenberger, G. Meynet, S. Ekström, C. Georgy, A. Maeder, C. Charbonnel, and L. Eyer. Stellar mass and age determinations. I. Grids of stellar models from $Z = 0.006$ to 0.04 and $M = 0.5$ to 3.5 $M_{\odot}$. *Astronomy & Astrophysics*, 541:A41, 2012. [p114]

A. Pietrinferni, S. Cassisi, M. Salaris, and F. Castelli. A large stellar evolution database for population synthesis studies. II. Stellar models and isochrones for an α-enhanced metal distribution. *Astrophysical Journal*, 642(2):797–812, 2006. [p108]

Matteo Dell'Omodarme
Dipartimento di Fisica "Enrico Fermi", Università di Pisa
Largo Pontecorvo 3, Pisa I-56127
Italy
mattdell@fastmail.fm

Giada Valle
Dipartimento di Fisica "Enrico Fermi", Università di Pisa
Largo Pontecorvo 3, Pisa I-56127
Italy
valle@df.unipi.it

Let Graphics Tell the Story - Datasets in R

by Antony Unwin, Heike Hofmann and Dianne Cook

Abstract Graphics are good for showing the information in datasets and for complementing modelling. Sometimes graphics show information models miss, sometimes graphics help to make model results more understandable, and sometimes models show whether information from graphics has statistical support or not. It is the interplay of the two approaches that is valuable. Graphics could be used a lot more in R examples and we explore this idea with some datasets available in R packages.

Introduction

One of the attractive features of R and its package system is that it allows the use of real datasets in the examples and vignettes to demonstrate statistical analyses. While we usually associate R with statistical methodology, it is hard to underestimate its impact as a data archive and dissemination tool. R has long ago overtaken data archives such as StatLib (Vlachos and Meyer, 1989 –) or OzDASL (Smyth, 2011) in scope and users. There are over 3000 data objects included in the 1001 packages available on CRAN, that have been submitted just before or after the release of R 2.15.2 (Trick or Treat) on Oct 26 2012. Not all of the objects are datasets in the traditional sense, that we deal with a two dimensional object with variables in the columns and records in the rows, but we were able to extract information on the numbers of columns and rows for 2428 of them. Figure 1 gives a summary of the number of packages and the number of associated datasets. A table of all packages and associated data objects can be found at `http://www.public.iastate.edu/~hofmann/data_in_r_sortable.html`.

For some of these datasets, graphics are also included, though the emphasis is usually more on modelling and analysis. In this article we want to show the value of including more graphics in R examples and to illustrate how they complement modelling.

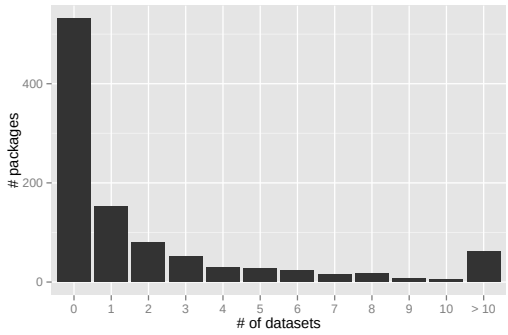


Figure 1: Basic summary of the number of datasets included in packages released on CRAN between Oct 5 and Nov 7 2012. The 1001 packages considered contain 3289 data objects in total.

Graphics help identify data quality issues; they reveal dataset structure and support the interpretation of results. The examples given here are taken from various packages and are more of a convenience sample than a random sample. In each case we give a brief description of the dataset, a reference to the dataset description in R, and sometimes some additional relevant information. We offer a selection of graphics, each one pointing out some feature of the dataset, and the code to produce them. Given the power of R's graphical tools and how easy it is to draw graphics, there is no need to restrict ourselves to a single all-encompassing display for a dataset. Instead, we recommend drawing many graphics to ensure that most, if not all, aspects of the dataset are presented. This approach takes up a lot of space in a printed copy, but is easy to use on a computer screen or a webpage.

We would like to see more use made of graphics with R, though only if each additional graphic conveys additional information. There is no value in drawing more graphics just because we can; they all have to tell a story. For each of the graphics in this article we have attempted to explain what particular information can be gathered from it. Sometimes there is overlap, in that the same information may be visible in more than one graphic. Also there may be alternative graphics readers might prefer to have drawn for presenting the same information. Graphics will always be very much a matter of taste and experience. As long as we are prepared to consider many graphics, we should be able to find good ones, which convey the information we want to draw attention to effectively.

Of course, graphics alone are not enough, just as analysis alone is rarely enough. We do not discuss any modelling here, as we concentrate on what graphics can contribute. Occasionally, we do point

out where modelling would be especially useful in checking or confirming an idea discovered from a graphic. In many cases a suitable model is obvious; in some others it is uncertain if there is any model that would do. Graphical displays are able to draw our attention to unusual features in a dataset, which we can recognise as being interesting without being clear how to test for them.

Examples

Anorexia - MASS

This dataset is used in the **MASS** package (Venables and Ripley, 2002) to illustrate a linear model fit, using `glm`. No plot is given with the example code. The data is also used in many other packages such as **granova** (Pruzek and Helmreich, 2010) where some complex plots are provided. The original source is Hand et al. (1993) who recommend "whichever statistical technique is employed, it is instructive to look at the three scatterplots of after/before". Curiously, none of the R users of the data follow this advice.

The dataset contains pre- and post-treatment weights for 72 young girls being treated for anorexia. Three treatment groups are considered: cognitive behavioural therapy (CBT), family therapy (FT) and a control group (Cont). We suggest adding code to the help page for the anorexia dataset (`?anorexia`) starting with:

```
data(anorexia, package = "MASS")
require("ggplot2")
qplot(data = anorexia, Treat, xlab = "", ylab = "")
```

This results in a barchart of the numbers in the three groups shown in Figure 2. This plot, as all of our plots, is drawn with the packages **ggplot2** (Wickham, 2009) and **vcd** (Meyer et al., 2006) and weaved into the document with **knitr** (Xie, 2012).

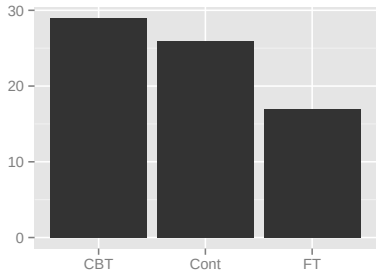


Figure 2: Barchart of the numbers of girls in each group in the anorexia study.

> *Story of Figure 2*: This plot shows 'just' data structure, but surprisingly the three groups are not of equal size. This may mean that some girls dropped out or that it was an observational study without random allocation of the participants, or there could be another explanation altogether. This should be borne in mind in interpreting the results.

Figure 3 shows – as originally suggested by Hand et al. (1993) – scatterplots of the weights of the girls in the three treatment groups. A simple version of this plot can be achieved with the code below:

```
## Suggested addition for ?anorexia (Cont.)
## - Simple
qplot(Prewt, Postwt, data=anorexia, colour=Treat, facets=.~Treat) +
  xlim(c(68,105)) + ylim(c(68,105))
```

A more polished variation also includes guide lines and leads the following code and Figure 3.

```
## - Polished
limits <- with(anorexia, range(c(Prewt, Postwt)))
limits <- limits + 0.05*c(-1,1)*limits

ggplot(data=anorexia, aes(x=Prewt, y=Postwt, colour=Treat)) +
  coord_equal(ylim=limits, xlim=limits) +
```

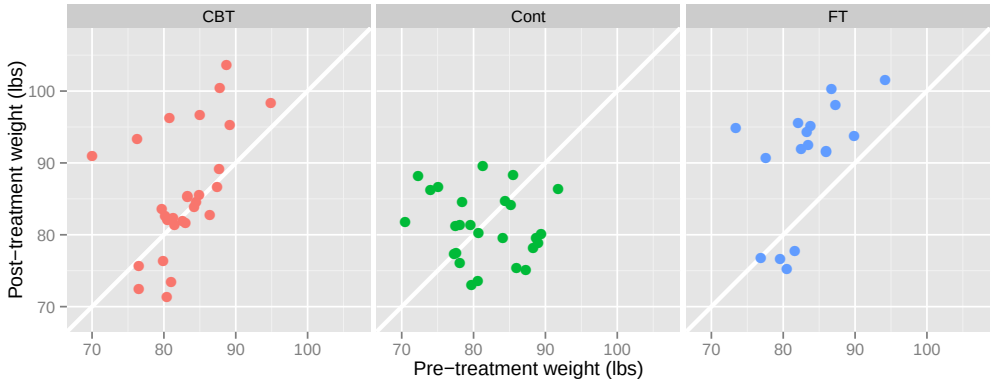


Figure 3: Scatterplots of pre- and post-treatment weights for the girls being treated for anorexia. $x = y$ guide lines have been added to help identify girls who gained or lost weight.

```
xlab("Pre-treatment weight (lbs)") +
ylab("Post-treatment weight (lbs)") +
geom_abline(intercept=0, slope=1, colour="white", size=1.25) +
geom_point(size=3) +
facet_grid(.~Treat) +
scale_colour_discrete(guide="none")
```

> *Story of Figure 3*: This plot gives a preliminary overview of the data. We learn that there were differences in the weight gains between the treatment groups. Girls in the control group tended to have lower post-weight, and roughly half lost weight. Girls in the family treatment group fall into two clusters: girls in one group tended to respond well to this treatment and gained weight, but for a small group of four girls this treatment was not successful. Similarly, there appear to be two groups of girls in the CBT group: a group of about eight girls responded well to the treatment, but a larger group of girls did not respond well.

A general linear model to fit post-treatment weight given pre-treatment weight is suggested in the existing help file of the function `glm`, as shown below. To this, we add graphical support as shown in Figure 4, which displays plots that support a linear model with change in weight plotted against pre-treatment weight. A guideline is drawn at zero. Noticeable differences can be seen between the treatments. This results in the following overall code:

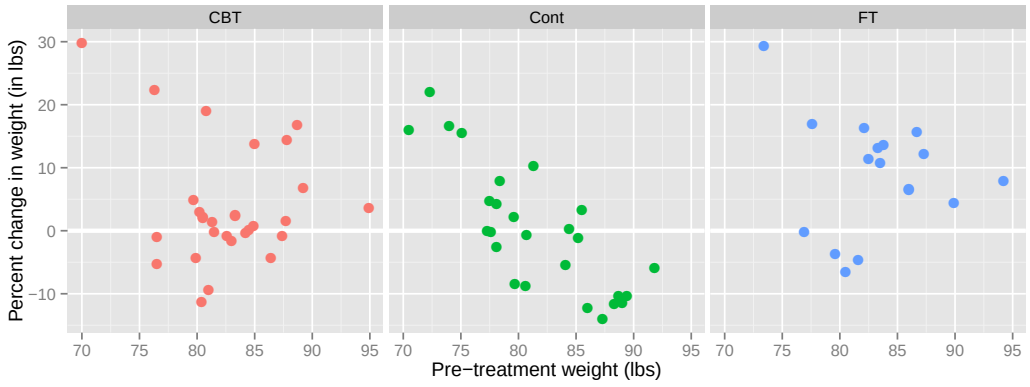


Figure 4: Percent change in weight by pre-treatment weight.

```
## Existing example code
?glm
anorex.1 <- glm(Postwt ~ Prewt+Treat+offset(Prewt), family = gaussian, data = anorexia)
summary(anorex.1)
```

```
## Suggested addition
ggplot(data=anorexia,
  aes(x=Prewt, colour=Treat, y=(Postwt-Prewt)/Prewt*100)) +
  xlab("Pre-treatment weight (lbs)") +
  ylab("Percent change in weight (in lbs)") +
  geom_hline(yintercept=0, size=1.25, colour="white") +
  geom_point(size=3) +
  facet_grid(.~Treat)+
  scale_colour_discrete(guide="none")
```

Story of Figure 4: There are differences in the weight gain between the treatments, but there is also a lot of individual variation. For some girls the two treatment methods clearly did not help them to gain weight. More girls in the control group failed to gain weight – only the very low weight girls gained much weight, which suggests that other influences may have been at work for these girls. The original data description indicates that intervention methods were used for the lightest girls. Generally there appears to be an association that the heavier girls, all still under normal weight, gained less weight than the lighter girls.

Various decisions were made in producing these plots to make them more informative.

- A raw data overview plot shows the data "straight out of the box". Such plots help the reader to identify any major issues with the data, such as extreme values or clusters.
- Colour was used to draw attention to the primary comparison of distributions of treatments. Facetting was chosen over plotting on the same plot because of the considerable variation within treatments, which makes it difficult to evaluate differences. Note that facetting, lattice structures (Sarkar, 2008), or trellis plots (Becker et al., 1996) all describe the process of positioning plots of different, usually mutually exclusive, subsets of the data next to each other. The subsetting is usually done by conditioning on co-variates. Conditioning has a central role in statistical data analysis, and finds its visual expression here.
- Guides for assisted viewing may be added, in this case the x = y line to determine which girls gained or lost weight. In other situations we could also employ data driven guides, such as contours, to focus attention on patterns.
- The restructured data plot shows the data as a support for the model. Although weight change is not the precise match to the response variable, the model is implicitly exploring this. It also allows reading of the weight change in a vertical direction instead of the off-diagonal view in the raw data plot. The horizontal guide line at zero allows the reader to see which girls gained weight. Using a scatterplot enables the reader to see the nuances of the data, such as clusters of girls who do well in one treatment, or fail to respond, and the indication that intervention may have been applied to the very underweight girls.

Lanza - HSAUR2

The `Lanza` data (Everitt and Hothorn, 2006; Lanza, 1987) consists of four clinical studies on the use of the drug Misoprostol to prevent gastrointestinal damage. Damage was measured as an ordinal response `classification` with levels 1 to 5 corresponding to a specified number of haemorrhages (with 1 being the fewest and 5 the most). The visualisation given in `?Lanza` is a set of four mosaicplots (without obvious information on which study is shown) shaded by residuals for an independence model.

Rather than a modelling result, we suggest the use of barcharts as a first introduction to the data. A simple barchart of study by treatment (not shown here) shows that all studies were balanced and that studies I to III had comparable numbers of patients, whereas study IV was smaller with just less than half the number of participants of the other studies. Breaking down the data further by the damage scores gives the multiple barcharts shown in Figure 5, requiring a single line extension of the existing code as shown below. We are making use of the small multiples paradigm (Tufte, 1983; Becker et al., 1996) to show study outcome in terms of number of haemorrhages in subsets by clinical trial and treatment:

```
## Existing code
data(Lanza, package="HSAUR2")

## Suggested example code for ?Lanza
qplot(classification, geom="bar", facets=treatment~study, data=Lanza)
```

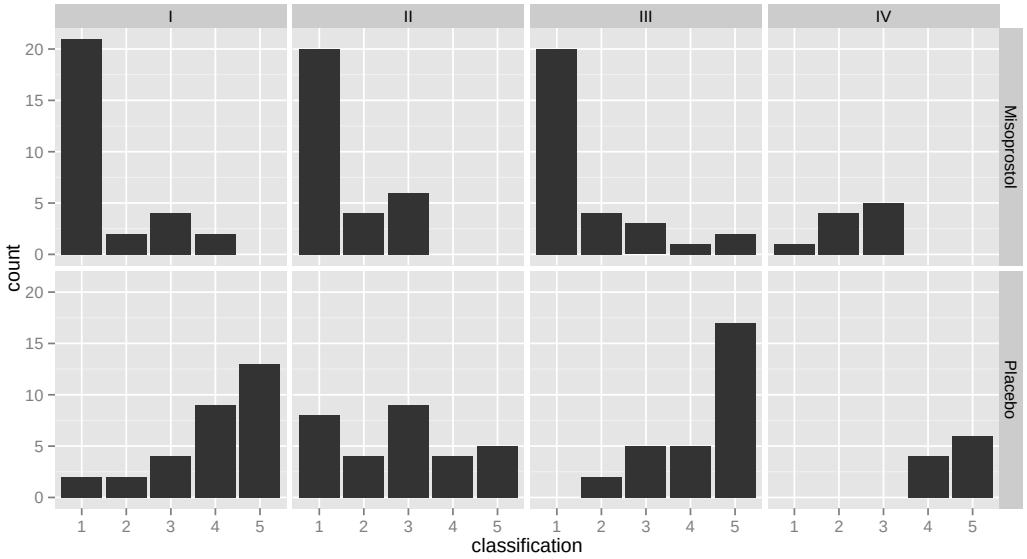


Figure 5: Number of haemorrhages classified from 1 (= best) to 5 (= worst) for patients in four clinical trials (left to right) treated with Misopropol (top row) or Placebo (bottom row).

> *Story of Figure 5*: All of the barcharts suggest that Misopropol is an effective drug for controlling or preventing haemorrhaging - in every study patients in the active treatment group reported better end results than the Placebo group. The distribution of these results was very similar for studies I and III, with the bulk of people in the Misopropol group being classified as '1', and the bulk of patients in the Placebo group classified as '5'. Study II showed a larger spread and less structure in the results for the Placebo group, while study IV showed the strongest distinction between results for the two treatment groups. Patients in treatment group IV also showed a different pattern of resulting haemorrhaging. Even in the active treatment group, the most frequent response was '3' - which might indicate, together with the relatively low number of patients, that study IV was done for a different target population with possibly more affected individuals.

Figure 6 is closer to the modelling approach suggested by the existing help file. The code for showing this figure is:

```
## Suggested addition for ?Lanza (Cont.)
levels(Lanza$treatment) <- c("P", "M")

require(vcd)
colors <- c("grey80", "darkred")
mosaic(treatment ~ classification | study, highlighting_fill=colors, data=Lanza)
```

> *Story of Figure 6*: A mosaic shows estimates of the conditional probability of being in the active treatment group given a particular haemorrhaging classification was reported. Since we know from our initial overview that treatment groups were balanced, this conditional probability is proportional to the conditional probability of seeing a specific outcome given treatment group, which is what we are more interested in, given the setup of the study. Again, we see that all four studies suggest a high effectiveness of Misopropol in terms of preventing haemorrhaging, since the presence of patients treated with Misopropol decreases with an increase of haemorrhages. Again, the different number of patients in study IV is visible (from the lower height of the rectangles in the last row of the plot), as well as its stronger separation of haemorrhaging outcomes between treatment groups.

schizophrenia2 - HSAUR2

The `schizophrenia2` data are from a ten-month longitudinal study following 44 individuals suffering from episodes of schizophrenia. Every two months (except at 4 months after the study began)

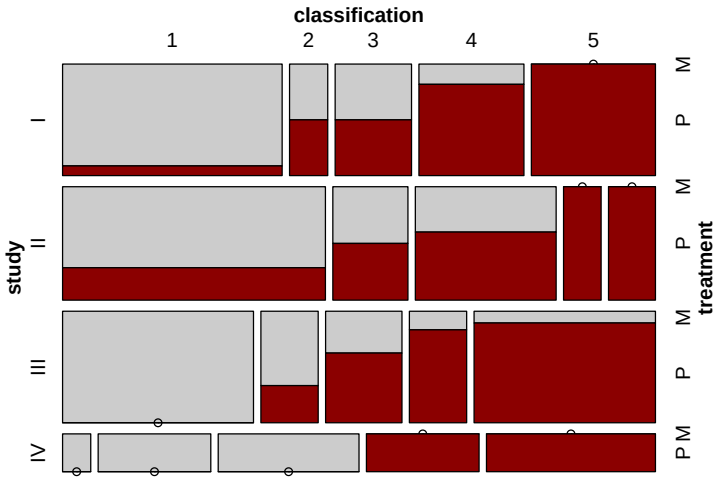


Figure 6: Mosaicplot of the `Lanza` Data: each row corresponds to one of the four clinical trials. Classification of haemorrhaging is shown from left to right. Patients in the Misopropol group (active treatment) are shown in red, patients in the Placebo group are shown in grey.

individuals reported whether the disorder was in an active state (`present`) or dormant (`absent`). Additionally, the data contain information about age at onset of the disease: under the age of 20 or, less often, over the age of 20.

The current help file presents a mosaicplot as an overview of the data, which we basically agree with, albeit we would like to suggest changes to the default values to emphasise various features in the data:

```
## Existing code
data("schizophrenia2", package = "HSAUR2")
mosaicplot(xtabs( ~ onset + month + disorder, data = schizophrenia2))

## suggested changes to the mosaic defaults:
levels(schizophrenia2$disorder) <- c("A", "P")
mosaic(disorder ~ month | onset, highlighting_fill = colors, data = schizophrenia2)
```

Changes to the defaults are based on recognition principles: reducing the spacing between rectangles emphasises grouping of bins and thereby helps with cognition of comparable subsets (Ware, 2004). Similarly, the pre-attentive property of colour (Healey, 2009) is used to give a strong additional visual cue of similarity. The change from vertical to horizontal layout emphasizes the longitudinal aspect of the study; by making the time component 'month' the x axis, we adhere to the convention of time-series plot.

> *Story of Figure 7*: This plot gives an overview of the `schizophrenia2` data. For both early onset (top row) and late onset of the disease (bottom row) the state of the disease is shown for every two months after the beginning of the study - with the inexplicable exception of 4 months after the study began. Individuals reporting the disease are coloured dark red, the remaining cases are shown in grey. As the study progressed, fewer individuals experienced episodes of schizophrenia. The slight shifts in vertical alignments of rectangles between onset classifications (upper and lower row of the mosaic) hint at the presence of missing values.

The next plot incorporates these missing values, but leaves everything else unchanged. The code for that is given as:

```
## Additional Suggestions for ?schizophrenia2
sc <- schizophrenia2
sc$disorder <- factor(sc$disorder, exclude=NULL)
levels(sc$disorder)[3] <- "dropout"
```

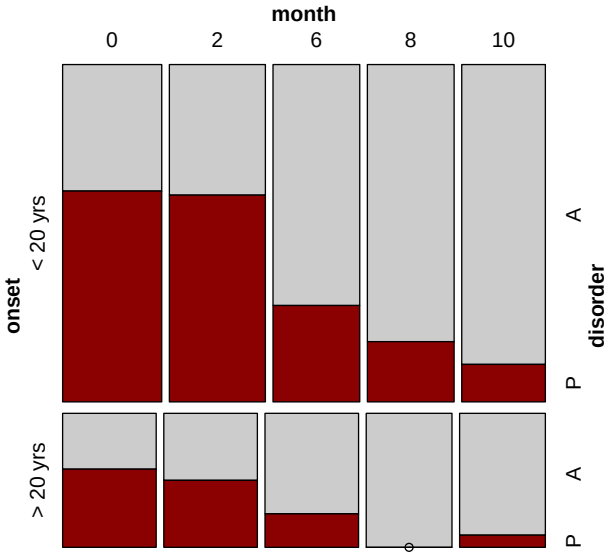


Figure 7: Mosaicplot of the `schizophrenia2` Data: 44 female patients were tracked over 10 months (from left to right) after hospitalisation, their disorder status is recorded as absent or present (grey versus dark red). Top and bottom row of the mosaic correspond to early onset of the disease (top row) versus onset after 20 years of age (bottom row).

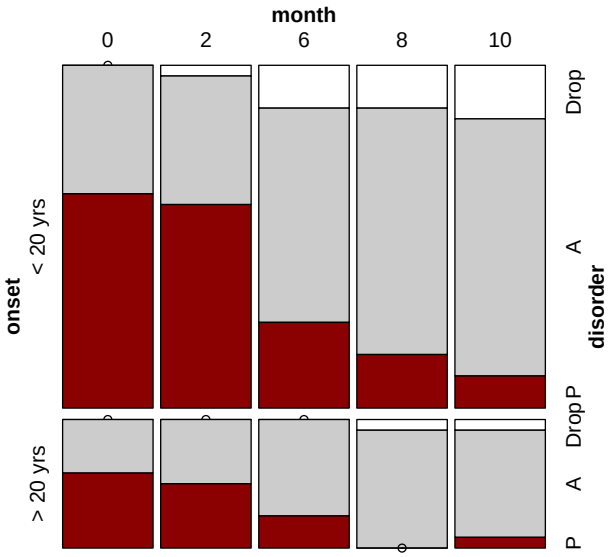


Figure 8: Mosaicplot of the `schizophrenia2` Data: the setup of the plot is the same as in Figure 7, but drop-out patients are now emphasised (white rectangles). Drop-out rates increased as the study progressed. Note that rectangles between different onset times of the disease are now aligned vertically.

```
sc$disorder <- factor(sc$disorder, levels=rev(c("present", "absent", "dropout")))

colors <- c("white", "grey80", "darkred")
mosaic(disorder ~ month | onset, highlighting_fill = colors, data = sc)
```

Story of Figure 8: Figure 8 emphasizes the presence of drop-outs from the study: the number of drop-outs increased as the study progressed. Drop-out rates between different onsets of the disease can be made comparable by changing the plot to a Doubledecker display using the parameter `direction=c("v","v","h")` in the code above (not shown).

Figure 9 summarizes individuals' experience throughout the study. Again, a mosaicplot is our suggested visualization. Ordering of the individuals is crucial for the plot and requires a surprising amount of coding to achieve the display we want. Sorting is an important, yet underestimated and underused tool in statistical graphics.

The following code chunk leads through all necessary steps necessary for producing Figure 9:

```
## Further Suggestions for ?schizophrenia2

## Calculate summary statistics of number and timing of episodes
## for each individual
require(plyr)
sc.present <- subset(sc, disorder == "present")

attacks <- ddply(sc.present,  .(subject),  summarise,
    episodes=length(month),
    first=min(month),
    last=max(month)
)

## Dropout information for each individual
sc.dropout <- subset(sc, disorder=="dropout")
drops <- ddply(sc.dropout,
  .(subject), summarise,
    drops=length(month)
)

## Merge all the information
sc <- merge(sc, drops, by="subject", all.x=T)
sc <- merge(sc, attacks, by="subject", all.x=T)
sc[is.na(sc)] <- 0

## hierarchical ordering:
## sort according to
##   number of episodes,
##   first episode,
##   last episode,
##   number of dropouts,
sc$subject <- reorder(sc$subject, sc$drops)
sc$subject <- reorder(sc$subject, sc$last)

sc$subject <- reorder(sc$subject, sc$first)
sc$subject <- reorder(sc$subject, sc$episodes)

sc.table <- with(sc, table(onset,  month, subject, disorder))
dimnames(sc.table)$subject <- rep("", 44)

par(mar=c(2,2,0,0))
## Finally, plot
mosaicplot(sc.table, dir=c("h","v","h","h"), col=c("white", "grey","darkred"),
           off=c(2,0,1,0), main="")
```

Story of Figure 9: The mosaic plot of Figure 9 shows each scheduled reporting of disease status for each individual in a box coloured according to the disease status (red for present,

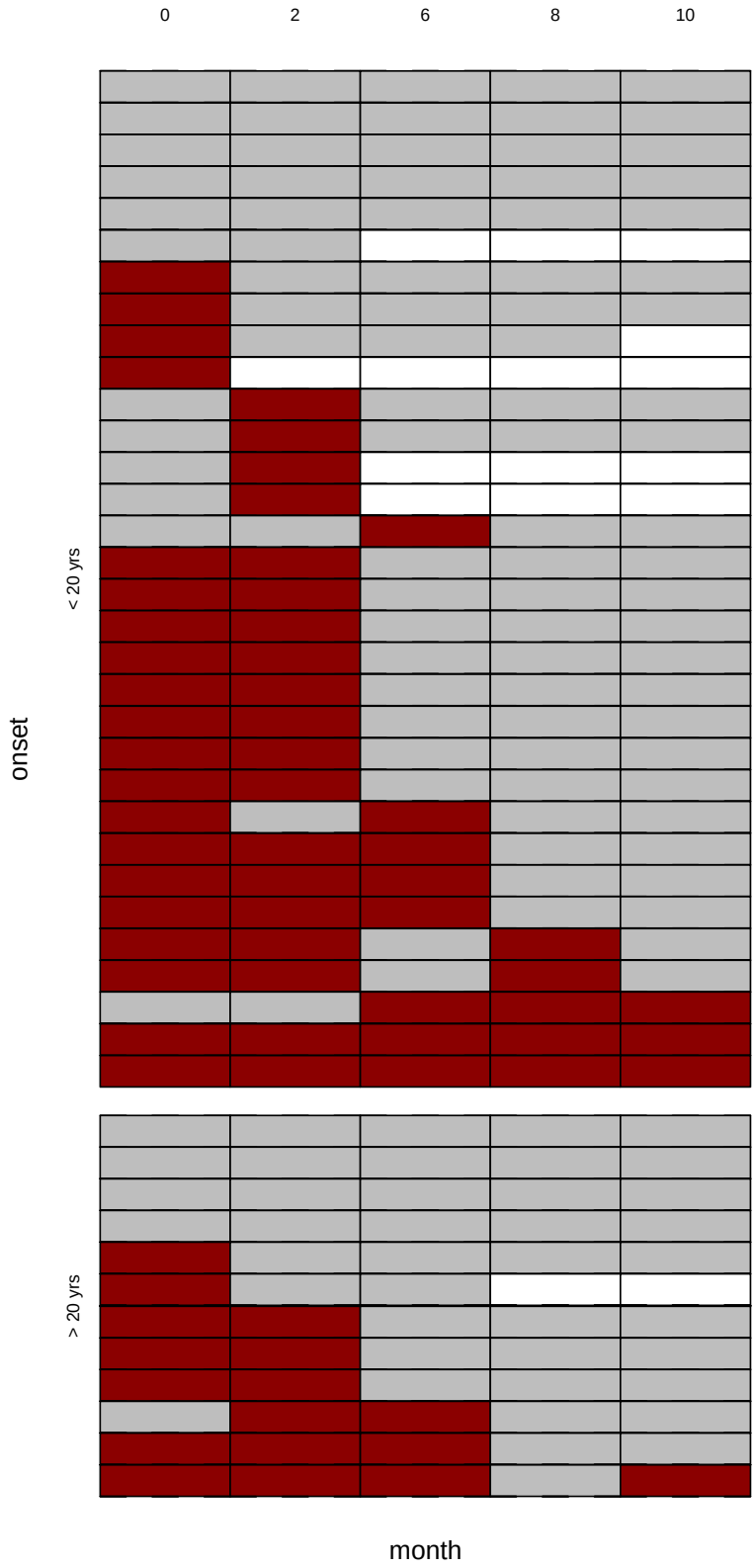


Figure 9: Mosaicplot of the `schizophrenia2` data focussing on individuals' experience throughout the study. Individuals are ordered according to the numbers of times absence or presence of the disease were reported.

grey for absent, white for not reported). Time in months is on the horizontal axis and each individual is represented in a horizontal row of boxes. In this setting, we distinguish early onset (top rows) versus late onset of the disease. Within each of these groups, subjects are ordered hierarchically according to number of episodes reported, first reported episode, last reported episode and lastly, number of non-reports. What becomes apparent is a pattern in the non-reports: once a person does not report for the first time, consecutive reports are missing as well. Additionally we see that the majority of participants do not suffer episodes at 8 or 10 months, if they did not report an episode beforehand. The similarity of reports is also remarkable; ignoring non-reports, there are potentially $2^5 = 32$ different patterns of reports - only 12 of these patterns show up among the 44 participants.

SexualFun – vcd

The `SexualFun` dataset contains responses of 91 married couples to the questionnaire item "Sex is fun for me and my partner". Each individual reports on a scale of "Always fun", "Very Often", "Fairly Often", and "Never Fun". The only visualization in the current version of `?SexualFun` is an agreement plot (Friendly, 2000; Meyer et al., 2010), which is based on Bangdiwala agreement statistics. As a precursor to that, we suggest the use of multiple barcharts, as shown in Figure 10, resulting from the code below, or a fluctuation diagram (Hofmann, 2000).

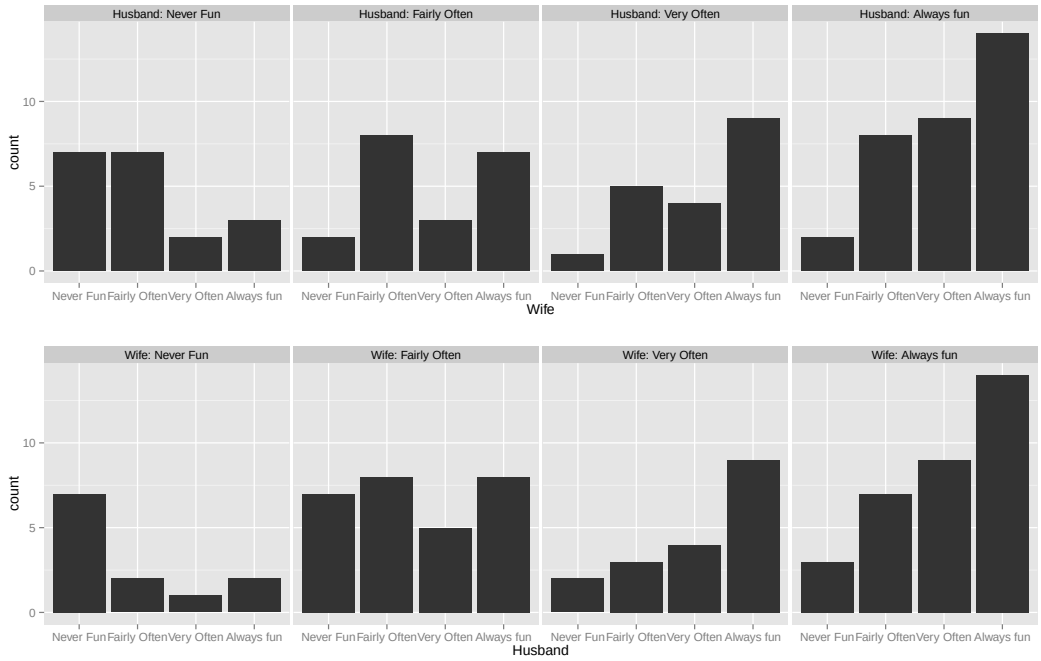


Figure 10: Barcharts of `SexualFun` showing the conditional distributions of responses to the questionnaire item “Sex is fun for me and my partner", conditioning first on the husband’s responses, then on the wife’s.

```
data(SexualFun, package="vcd")

# Suggested example code for ?SexualFun
sf  <- as.data.frame(SexualFun)

qplot(Wife, data=sf, weight=Freq) + facet_grid(facets=.~Husband, labeller="label_both")
qplot(Husband, data=sf, weight=Freq) + facet_grid(facets=.~Wife, labeller="label_both")
```

Story of Figure 10: The conditional barcharts show that the distributions are very similar except for the combinations “Never Fun" and “Fairly Often", where the wives say “Never Fun" less frequently. Frequencies of "Always fun" are approximately equally high for both husbands and wives. In general, responses are spread across all possible combinations, with larger differences between couples occurring less frequently.

Assuming integer scores 1 to 4 for levels “Never Fun" to “Always fun", we can compute a joint score for each of the couples as the sum of their answers and an index of disagreement as the difference between husbands' and wives' scores. Plotting this is similar to a Tukey transformation (Tukey, 1977), i.e. a rotation of a fluctuation diagram by 45 degrees.

This is achieved by the code below. Note that the command `ggfluctuation` is deprecated in the current version of **ggplot2**, but still works locally (after removing the deprecation notice).

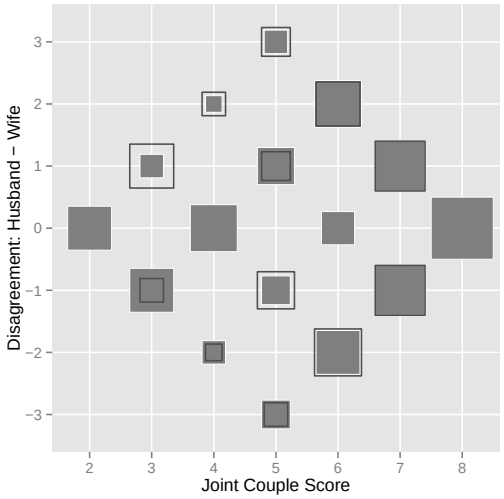


Figure 11: Rotated Fluctuation Diagram of `SexualFun`: disagreement of the partners is plotted on the y axis against the joint couple score on the x axis. Black rectangles are overlaid – these show the mirror image along the horizontal axis to emphasize differences in husbands' and wives' scores.

```
## Tukey Transformation:
## rotation by 45 degree
sf$husband <- as.numeric(sf$Husband)
sf$wife <- as.numeric(sf$Wife)

sf$Scale <- with(sf, husband+wife)
sf$Disagree <- with(sf, husband-wife)

## diamond shaped fluctuation diagram
sf$range <- sqrt(sf$Freq)
sf$range <- sf$range/max(sf$range)/2

ggfluctuation( xtabs(Freq~Disagree+Scale, data=sf)) +
 xlab("Joint Couple Score") +
 ylab("Disagreement: Husband - Wife") +
 geom_rect(aes(xmin=Scale-1-range, ymin=-Disagree+4-range,
               xmax=Scale-1+range, ymax=-Disagree+4+range),
          fill=NA, colour="grey30", data=subset(sf, Disagree!=0), inherit.aes=FALSE)
```

Story of Figure 11: This rotated fluctuation diagram emphasizes the numbers of agreements and disagreements among the couples directly. Symmetry above the middle line and asymmetry below reveal agreement and disagreement between husbands and wives. The fluctuation diagram is overlaid by outlines of the other spouses' squares to highlight deviation from symmetry. The top left corner again confirms what we have already observed: fewer women than men responded with “Never fun".

Figures 10 and 11 show the same information, though in Figure 10 each data point is shown twice. Figure 11 is more efficient but requires more work to decipher. With the overlaid rectangles it is in particular helpful to highlight asymmetry in the answers between husbands and wives. There are many more alternatives to the plots that we have chosen. One alternative to the set of barcharts would be stacked barcharts, for instance the likert plots implemented in Richard Heiberger's **HH** package (Heiberger, 2012). Interactive graphics would be beneficial to both.

Discussion and conclusions

What do we want? We would like to ask R Core to add a recommendation in the existing test suite for packages to show plots within the example section of a dataset documentation. We don't ask for this lightly, and it certainly is not an ideal situation, since it adds yet another hurdle that developers need to pass on their way to creating and submitting a package on CRAN — nor does it guarantee that the plots suggested would be very helpful. However, this falls in line with standard rules for CRAN — checks are not based on content, but are syntactical. This is done in the hope, that syntactic violations, for which we can create automatic tests, are correlated with offenses in terms of content. From a technical point of view the infrastructure of the package system is already there. A plotting recommendation for datasets is feasible without introducing a massive overhead because of dependencies on various graphics packages: the `Suggests` specification in the description file takes care of exactly those situations. It allows the listing of all packages that do not need to be loaded for a package, but are necessary for successfully running the examples.

Data is at the heart of our discipline. Having stricter rules on data descriptions will give data a more central position in the R community. The absence of good data documentations, both in terms of the data structure and potential findings, devalues its usefulness. Documentation is a large part of achieving reproducibility of research results, which has been one of R's goals (Gentleman and Temple Lang, May 2004) for a long time. Past (Leisch, 2002) and recent advances (Xie, 2012) in tools for reproducible research have facilitated the use of the existing infrastructure so much that we should more actively employ it.

Help files do not currently include any output, neither graphics nor any model output, and that is probably a good thing. Users can simply run the code and see for themselves what happens. What we are asking for are some pieces of code in the help files for creating (sensible) graphics along the lines shown in this paper.

All our examples share the same unifying themes. Simple initial graphics give a good overview of basic information — like the sizes of treatment groups and potential differences between them — and identify problems with the data — such as data errors, outliers or heaping. Graphics can enhance data analyses by suggesting new angles and by illustrating results. Graphics have to tell a story just as any statistical model must and stories should be given explicitly in the text to the graphics. Stories in graphics are rarely as "obvious" or "easy to see" as is often claimed. From a plot creator's point of view this gives us also a way to check if a plot is worth a story — if we are having a hard time telling the story, we might want to take this as a hint, that we should change the plot to something that is easier to talk about.

Presenting a collection of graphics is usually better than trying to convey all of a dataset's information in a single display.

Bibliography

R. A. Becker, W. S. Cleveland, and M. J. Shyu. The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, 5(2):123–155, 1996. [p120]

B. S. Everitt and T. Hothorn. *HSAUR2: A Handbook of Statistical Analyses Using R*. Chapman & Hall/CRC, 2nd edition, 2006. URL http://CRAN.R-project.org/package=HSAUR2. R package version 1.0-2. [p120]

M. Friendly. *Visualizing Categorical Data*. SAS Publishing, 2000. [p126]

R. Gentleman and D. Temple Lang. Statistical analyses and reproducible research. Bioconductor Project Working Papers, May 2004. [p128]

D. J. Hand, F. Daly, K. McConway, D. Lunn, and E. Ostrowski. *A handbook of small datasets*. Chapman & Hall/CRC, 1993. [p118]

C. Healey. Perception in visualisation, 2009. URL http://www.csc.ncsu.edu/faculty/healey/PP/index.html. [p122]

R. M. Heiberger. *HH: Statistical Analysis and Data Display: Heiberger and Holland*, 2012. URL http://CRAN.R-project.org/package=HH. R package version 2.3-17. [p127]

H. Hofmann. Exploring categorical data: Interactive mosaic plots. *Metrika*, 51(1):11–26, 2000. [p126]

F. Lanza. A double-blind study of prophylactic effect of misoprostol on lesions of gastric and duodenal mucosa induced by oral administration of tolmetin in healthy subjects. *British Journal of Clinical Practice*, pages 91–101, 1987. [p120]

F. Leisch. Dynamic generation of statistical reports using literate data analysis, 2002. [p128]

D. Meyer, AchimZeileis, and K. Hornik. The strucplot framework: Visualizing multi-way contingency tables with vcd. *Journal of Statistical Software*, 17(3):1–48, 2006. URL http://www.jstatsoft.org/v17/i03/. [p118]

D. Meyer, A. Zeileis, and K. Hornik. vcd: Visualizing categorical data., 2010. [p126]

R. M. Pruzek and J. E. Helmreich. *granova: Graphical Analysis of Variance*, 2010. URL http://CRAN.R-project.org/package=granova. R package version 2.0. [p118]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. URL http://lmdvr.r-forge.r-project.org. ISBN 978-0-387-75968-5. [p120]

G. K. Smyth. Australasian Data and Story Library (OzDASL), 2011. URL http://www.statsci.org/data. [p117]

E. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983. [p120]

J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977. [p127]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0. [p118]

P. Vlachos and M. Meyer. StatLib: Data, Software and News from the Statistics Community, 1989 –. URL http://lib.stat.cmu.edu/. [p117]

C. Ware. *Information Visualization*. Morgan Kaufmann, San Francisco, CA, USA, 2nd edition, 2004. [p122]

H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book. [p118, 199]

Y. Xie. *knitr: A general-purpose package for dynamic report generation in R*, 2012. URL http://CRAN.R-project.org/package=knitr. R package version 0.5. [p118, 128]

Antony R Unwin
Computer-Oriented Statistics And Data Analysis
Augsburg University
Germany
unwin@math.uni-augsburg.de

Heike Hofmann
Department of Statistics
Iowa State University
United States
hofmann@mail.iastate.edu

Dianne H Cook
Department of Statistics
Iowa State University
United States
dicook@mail.iastate.edu

Estimating Spatial Probit Models in R

by Stefan Wilhelm and Miguel Godinho de Matos

Abstract In this article we present the Bayesian estimation of spatial probit models in R and provide an implementation in the package **spatialprobit**. We show that large probit models can be estimated with sparse matrix representations and Gibbs sampling of a truncated multivariate normal distribution with the precision matrix. We present three examples and point to ways to achieve further performance gains through parallelization of the Markov Chain Monte Carlo approach.

Introduction

The abundance of geolocation data and social network data has lead to a growing interest in spatial econometric methods, which model a contemporaneous dependence structure using neighboring observations (e.g. friends in social networks).

There are a variety of R packages for spatial models available, an incomplete list includes **spBayes** (Finley and Banerjee, 2013), **spatial** (Venables and Ripley, 2002), geostatistical packages called **geoR** (Diggle and Ribeiro, 2007) and **sgeostat** (Majure and Gebhardt, 2013), **spdep** (Bivand, 2013), **sphet** (Piras, 2010), **sna** (Butts, 2013) and **network** (Butts et al., 2013).

While all of these packages deal with linear spatial models, in this article we focus on a nonlinear model, the spatial probit model, and present the Bayesian estimation first proposed by LeSage (2000). As will be shown below, one crucial point we have been working on was the generation of random numbers of a truncated multivariate normal distribution in very high dimensions.

Together with these high dimensions and the size of problems in spatial models comes the need to work with sparse matrix representations rather than the usual dense `matrix()`. Popular R packages for dealing with sparse matrices are **Matrix** (Bates and Maechler, 2013) and **sparseM** (Koenker and Ng, 2013).

In the next section we first introduce spatial probit models and describe the Bayesian estimation procedure of the SAR probit model in detail. Subsequently we discuss some implementation issues in R, describe the `sarprobit` method in package **spatialprobit** (Wilhelm and de Matos, 2013), compare it against maximum likelihood (ML) and generalized method of moments (GMM) estimation in **McSpatial** (McMillen, 2013) and illustrate the estimation with an example from social networks. We also illustrate how to parallelize the Bayesian estimation with the **parallel** package.

Spatial probit models

The book of LeSage and Pace (2009) is a good starting point and reference for spatial econometric models in general and for limited dependent variable spatial models in particular (chapter 10, p. 279).

Suppose we have the spatial autoregressive model (SAR model, spatial lag model)

$$\mathbf{z} = \rho\mathbf{W}\mathbf{z} + \mathbf{X}\boldsymbol{\beta} + \epsilon, \quad \epsilon \sim N\left(0, \sigma_\epsilon^2 \mathbf{I}_n\right) \tag{1}$$

for $\mathbf{z} = (z_1, \ldots, z_n)'$ with some fixed matrix of covariates $\mathbf{X}$ $(n \times k)$ associated with the parameter vector $\boldsymbol{\beta}$ $(k \times 1)$. The matrix $\mathbf{W}$ $(n \times n)$ is called the spatial weight matrix and captures the dependence structure between neighboring observations such as friends or nearby locations. The term $\mathbf{W}\mathbf{z}$ is a linear combination of neighboring observations. The scalar ρ is the dependence parameter and will assumed $\text{abs}(\rho) < 1$. The $k+1$ model parameters to be estimated are the parameter vector $\boldsymbol{\beta}$ and the scalar ρ.

In a spatial probit model, $\mathbf{z}$ is regarded as a latent variable, which cannot be observed. Instead, the observables are only binary variables y_i (0, 1) as

$$y_i = \begin{cases} 1 & \text{if } z_i \geq 0, \\ 0 & \text{if } z_i < 0. \end{cases}$$

y_i can reflect any binary outcome such as survival, a buy/don't buy decision or a class variable in binary classification problems. For identification, σ_ϵ^2 is often set to $\sigma_\epsilon^2 = 1$.

The data generating process for $\mathbf{z}$ is

$$\begin{aligned}\mathbf{z} &= (\mathbf{I}_n - \rho\mathbf{W})^{-1}\mathbf{X}\boldsymbol{\beta} + (\mathbf{I}_n - \rho\mathbf{W})^{-1}\boldsymbol{\epsilon} \\ \boldsymbol{\epsilon} &\sim N(0, \mathbf{I}_n)\end{aligned}$$

Note that if $\rho = 0$ or $\mathbf{W} = \mathbf{I}_n$, the model reduces to an ordinary probit model, for which Wooldridge (2002, chapter 17) and Albert (2007, section 10.1) are good references. The ordinary probit model can be estimated in R by `glm()`.

Another popular spatial model is the spatial error model (SEM) which takes the form

$$\begin{aligned}&\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}, \quad \mathbf{u} = \rho\mathbf{W}\mathbf{u} + \epsilon, \quad \epsilon \sim N\left(0, \sigma_\epsilon^2\mathbf{I}_n\right) \\ &\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + (\mathbf{I}_n - \rho\mathbf{W})^{-1}\boldsymbol{\epsilon}\end{aligned} \tag{2}$$

where as before, only binary variables y_i (0,1) can be observed instead of z_i. Our following discussion focuses on the spatial lag probit model (1), but the estimation of the probit model with spatial errors is covered in **spatialprobit** as well.

Recent studies using spatial probit models include, among others, Marsh et al. (2000), Klier and McMillen (2008) and LeSage et al. (2011).

Bayesian estimation

Maximum Likelihood and GMM estimation of the spatial probit is implemented in the package **McSpatial** (McMillen, 2013) with the methods `spprobitml` and `spprobit`. Here we present the details of Bayesian estimation. Although GMM is massively more efficient computationally than Bayesian Markov Chain Monte Carlo (MCMC), as a instrumental-variables estimation it will work well only in very large samples. This point is also brought by Franzese et al. (2013), who compares different spatial probit estimation strategies.

The basic idea in Bayesian estimation is to sample from a posterior distribution of the model parameters $p(\mathbf{z}, \boldsymbol{\beta}, \rho|\mathbf{y})$ given the data $\mathbf{y}$ and some prior distributions $p(\mathbf{z})$, $p(\boldsymbol{\beta})$, $p(\rho)$. See for example Albert (2007) and the accompanying package **LearnBayes** for an introduction to Bayesian statistics in R (Albert, 2012). This sampling for the posterior distribution $p(\mathbf{z}, \boldsymbol{\beta}, \rho|\mathbf{y})$ can be realized by a Markov Chain Monte Carlo and Gibbs sampling scheme, where we sample from the following three conditional densities $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$, $p(\boldsymbol{\beta}|\mathbf{z}, \rho, \mathbf{y})$ and $p(\rho|\mathbf{z}, \boldsymbol{\beta}, \mathbf{y})$:

1. Given the observed variables $\mathbf{y}$ and parameters $\boldsymbol{\beta}$ and ρ, we have $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$ as a truncated multinormal distribution

$$\mathbf{z} \sim N\left((\mathbf{I}_n - \rho\mathbf{W})^{-1}\mathbf{X}\boldsymbol{\beta}, \left[(\mathbf{I}_n - \rho\mathbf{W})'(\mathbf{I}_n - \rho\mathbf{W})\right]^{-1}\right) \tag{3}$$

   subject to $z_i \geq 0$ for $y_i = 1$ and $z_i < 0$ for $y_i = 0$, which can be efficiently sampled from using the method `rtmvnorm.sparseMatrix()` in package **tmvtnorm** (Wilhelm and Manjunath, 2013), see the next section for details.

2. For a normal prior $\boldsymbol{\beta} \sim N(\mathbf{c}, \mathbf{T})$, we can sample $p(\boldsymbol{\beta}|\rho, \mathbf{z}, \mathbf{y})$ from a multivariate normal as

$$\begin{aligned}p(\boldsymbol{\beta}|\rho, \mathbf{z}, \mathbf{y}) &\propto N(\mathbf{c}^*, \mathbf{T}^*) \\ \mathbf{c}^* &= \left(\mathbf{X}'\mathbf{X} + \mathbf{T}^{-1}\right)^{-1}\left(\mathbf{X}'\mathbf{S}\mathbf{z} + \mathbf{T}^{-1}\mathbf{c}\right) \\ \mathbf{T}^* &= \left(\mathbf{X}'\mathbf{X} + \mathbf{T}^{-1}\right)^{-1} \\ \mathbf{S} &= (\mathbf{I}_n - \rho\mathbf{W})\end{aligned} \tag{4}$$

   The standard way for sampling from this distribution is `rmvnorm()` from package **mvtnorm** (Genz et al., 2013).

3. The remaining conditional density $p(\rho|\boldsymbol{\beta}, \mathbf{z}, \mathbf{y})$ is

$$p(\rho|\boldsymbol{\beta}, \mathbf{z}, \mathbf{y}) \propto |\mathbf{I}_n - \rho\mathbf{W}| \exp\left(-\frac{1}{2}(\mathbf{S}\mathbf{z} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{S}\mathbf{z} - \mathbf{X}\boldsymbol{\beta})\right) \tag{5}$$

   which can be sampled from using Metropolis-Hastings or some other sampling scheme (i.e. Importance Sampling). We implement a grid-based evaluation and numerical integration proposed by LeSage and Pace and then draw from the inverse distribution function (LeSage and Pace, 2009, p. 132).

Implementation issues

Since MCMC methods are widely considered to be (very) slow, we devote this section to the discussion of some implementation issues in R. Key factors to estimate large spatial probit models in R include the usage of sparse matrices and compiled Fortran code, and possibly also parallelization, which has been introduced to R 2.14.0 with the package **parallel**. We report estimation times and memory requirements for several medium-size and large-size problems and compare our times to those reported by LeSage and Pace (2009, p. 291). We show that our approach allows the estimation of large spatial probit models in R within reasonable time.

Drawing $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$

In this section we describe the efforts made to generate samples from $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$

$$\mathbf{z} \sim N\left((\mathbf{I}_n - \rho\mathbf{W})^{-1} \mathbf{X}\boldsymbol{\beta}, \left[(\mathbf{I}_n - \rho\mathbf{W})' (\mathbf{I}_n - \rho\mathbf{W}) \right]^{-1} \right)$$

subject to $z_i \geq 0$ for $y_i = 1$ and $z_i < 0$ for $y_i = 0$.

The generation of samples from a truncated multivariate normal distribution in high dimensions is typically done with a Gibbs sampler rather than with other techniques such as rejection sampling (for a comparison, see Wilhelm and Manjunath (2010)). The Gibbs sampler draws from univariate conditional densities $f(z_i|\mathbf{z}_{-i}) = f(z_i|z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)$.

The distribution of z_i conditional on $\mathbf{z}_{-i}$ is univariate truncated normal with variance

$$\begin{aligned} \boldsymbol{\Sigma}_{i.-i} &= \boldsymbol{\Sigma}_{ii} - \boldsymbol{\Sigma}_{i,-i}\boldsymbol{\Sigma}_{-i,-i}^{-1}\boldsymbol{\Sigma}_{-i,i} && (6)\\ &= \mathbf{H}_{ii}^{-1} && (7) \end{aligned}$$

and mean

$$\begin{aligned} \boldsymbol{\mu}_{i.-i} &= \boldsymbol{\mu}_i + \boldsymbol{\Sigma}_{i,-i}\boldsymbol{\Sigma}_{-i,-i}^{-1} \left(\mathbf{z}_{-i} - \boldsymbol{\mu}_{-i} \right) && (8)\\ &= \boldsymbol{\mu}_i - \mathbf{H}_{ii}^{-1}\mathbf{H}_{i,-i} \left(\mathbf{z}_{-i} - \boldsymbol{\mu}_{-i} \right) && (9) \end{aligned}$$

and bounds $-\infty \leq z_i \leq 0$ for $y_i = 0$ and $0 \leq z_i \leq \infty$ for $y_i = 1$.

Package **tmvtnorm** has such a Gibbs sampler in place since version 0.9, based on the covariance matrix $\boldsymbol{\Sigma}$. Following Geweke (1991, 2005, p. 171), the Gibbs sampler is easier to state in terms of the precision matrix $\mathbf{H}$, simply because it requires fewer and easier operations in the above equations (7) and (9). These equations will further simplify in the case of a sparse precision matrix $\mathbf{H}$, in which most of the elements are zero. With sparse $\mathbf{H}$, operations involving $\mathbf{H}_{i,-i}$ need only to be executed for all non-zero elements rather than for all $n-1$ variables in $\mathbf{z}_{-i}$.

In our spatial probit model, the covariance matrix $\boldsymbol{\Sigma} = [(\mathbf{I}_n - \rho\mathbf{W})'(\mathbf{I}_n - \rho\mathbf{W})]^{-1}$ is a dense matrix, whereas the corresponding precision matrix $\mathbf{H} = \boldsymbol{\Sigma}^{-1} = (\mathbf{I}_n - \rho\mathbf{W})'(\mathbf{I}_n - \rho\mathbf{W})$ is sparse. Hence, using $\boldsymbol{\Sigma}$ for sampling $\mathbf{z}$ is inefficient. For this reason we reimplemented the Gibbs sampler with the precision matrix $\mathbf{H}$ in package **tmvtnorm** instead (Wilhelm and Manjunath, 2013).

Suppose one wants to draw N truncated multinormal samples in n dimensions, where the precision matrix $\mathbf{H}$ is sparse $(n \times n)$ with only $m < n$ entries different from zero per row on average (e.g. $m = 6$ nearest neighbors or average branching factor in a network). With growing dimension n, two types of problems arise with a usual dense matrix representation for $\mathbf{H}$:

1. **Data storage problem:** Since matrix $\mathbf{H}$ is $n \times n$, the space required to store the dense matrix will be quadratic in n. One remedy is, of course, using some sparse matrix representation for $\mathbf{H}$ (e.g. packages **Matrix**, **sparseM** etc.), which actually holds only $n \cdot m$ elements instead of $n \cdot n$. The following code example shows the difference in object sizes for a dense vs. sparse identity matrix $\mathbf{I}_n$ for $n = 10000$ and $m = 1$.

```
> library(Matrix)
> I_n_dense <- diag(10000)
> print(object.size(I_n_dense), units = "Mb")

762.9 Mb

> rm(I_n_dense)
> I_n_sparse <- sparseMatrix(i = 1:10000, j = 1:10000, x = 1)
> print(object.size(I_n_sparse), units = "Mb")

0.2 Mb
```

2. **Data access problem:** Even with a sparse matrix representation for $\mathbf{H}$, a naive strategy that tries to access an arbitrary matrix element `H[i,j]` like in triplet or hash table representations, will result in $N \cdot n \cdot n$ matrix accesses to $\mathbf{H}$. This number grows quadratically in n. For example, for $N = 30$ and $n = 10000$ it adds up to $30 \cdot 10000 \cdot 10000 = 3$ billion hash function calls, which is inefficient and too slow for most applications.
   Iterating only over the non-zero elements in row i, for example in term $H_{i,-i}$, reduces the number of accesses to $\mathbf{H}$ to only $N \cdot n \cdot m$ instead of $N \cdot n \cdot n$, which furthermore will only grow linearly in n for a fixed m. Suitable data structures to access all non-zero elements of the precision matrix $\mathbf{H}$ are linked lists of elements for each row i (list-of-lists) or, thanks to the symmetry of $\mathbf{H}$, the compressed-sparse-column/row representation of sparse matrices, directly available from the **Matrix** package.

How fast is the random number generation for $\mathbf{z}$ now? We performed a series of tests for varying sizes N and n and two different values of m. The results presented in Table 1 show that the sampler with sparse $\mathbf{H}$ is indeed very fast and works fine in high dimensions. The time required scales with $N \cdot n$.

		N					
	n	10^1	10^2	10^3	10^4	10^5	10^6
m=2	10^1				0.03	0.25	2.60
	10^2			0.03	0.25	2.75	
	10^3		0.03	0.25	2.84		
	10^4	0.03	0.25	2.80			
	10^5	0.26	2.79				
m=6	10^1				0.03	0.23	2.48
	10^2			0.01	0.23	2.53	
	10^3		0.03	0.23	2.59		
	10^4	0.02	0.24	2.57			
	10^5	0.25	2.68				

Table 1: Performance test results for the generation of truncated multivariate normal samples $\mathbf{z}$ with `rtmvnorm.sparseMatrix()` (**tmvtnorm**) for a varying number of samples N and dimension n. The precision matrix in each case is $\mathbf{H} = (\mathbf{I}_n - \rho\mathbf{W})'(\mathbf{I}_n - \rho\mathbf{W})$, the spatial weight matrix $\mathbf{W}$ contains m non-zero entries per row. Times are in seconds and measured on an Intel® Core™i7-2600 CPU @3.40 GHz.

One more performance issue we discuss here is the burn-in size in the innermost Gibbs sampler generating $\mathbf{z}$ from $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$. Depending on the start value, Gibbs samplers often require a certain burn-in phase until the sampler mixes well and draws from the desired target distribution. In our MCMC setup, we only draw one sample of $\mathbf{z}$ from $p(\mathbf{z}|\boldsymbol{\beta}, \rho, \mathbf{y})$ in each MCMC iteration, but possibly in very high dimensions (e.g. $N = 1$, $n = 10000$). With `burn.in=20` samples, we have to generate 21 draws in order to keep just one. In our situation, a large burn-in size will dramatically degrade the MCMC performance, so the number of burn-in samples for generating $\mathbf{z}$ has to be chosen carefully. LeSage and Pace (2009) discuss the role of the burn-in size and often use `burn.in=10`, when the Gibbs sampler starts from zero. Alternatively, they also propose to use no burn-in phase at all (e.g. `burn.in=0`), but then to set the start value of the sampler to the previous value of $\mathbf{z}$ instead of zero.

QR decomposition of $(\mathbf{I}_n - \rho\mathbf{W})$

The mean vector $\boldsymbol{\mu}$ of the truncated normal samples $\mathbf{z}$ in equation (3) takes the form

$$\boldsymbol{\mu} = (\mathbf{I}_n - \rho\mathbf{W})^{-1}\mathbf{X}\boldsymbol{\beta} \tag{10}$$

However, inverting the sparse matrix $\mathbf{S} = (\mathbf{I}_n - \rho\mathbf{W})$ will produce a dense matrix and will therefore offset all benefits from using sparse matrices (i.e. memory consumption and size of problems that can be solved). It is preferable to determine $\boldsymbol{\mu}$ by solving the equations $(\mathbf{I}_n - \rho\mathbf{W})\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$ with a QR decomposition of $\mathbf{S} = (\mathbf{I}_n - \rho\mathbf{W})$. We point out that there is a significant performance difference between the usual code `mu <-qr.solve(S,X %*% beta)` and `mu <-solve(qr(S),X %*% beta)`. The latter will apply a QR decomposition for a sparse matrix $\mathbf{S}$ and will use a method `qr()` from the package **Matrix**, whereas the first function from the **base** package will coerce $\mathbf{S}$ into a dense matrix.

`solve(qr(S),X %*% beta)` takes only half the time required by `qr.solve(S,X %*% beta)`.

Drawing $p(\boldsymbol{\beta}|\mathbf{z}, \rho, \mathbf{y})$

Drawing $p(\boldsymbol{\beta}|\mathbf{z}, \rho, \mathbf{y})$ in equation (4) is a multivariate normal distribution, whose variance $\mathbf{T}^*$ does not depend on $\mathbf{z}$ and ρ. Therefore, we can efficiently vectorize and generate temporary draws $\beta_{tmp} \sim N(0, \mathbf{T}^*)$ for all MCMC iterations before running the chain and then just shift these temporary draws by $\mathbf{c}^*$ in every iteration to obtain a $\beta \sim N(\mathbf{c}^*, \mathbf{T}^*)$.

Determining log-determinants and drawing $p(\rho|\mathbf{z}, \boldsymbol{\beta}, \mathbf{y})$

The computation of log-determinants for $\ln|I_n - \rho W|$, whose evaluation is frequently needed for drawing $p(\rho|\mathbf{z}, \boldsymbol{\beta}, \mathbf{y})$, becomes a challenging task for large matrices. Bivand (2010) gives a survey of the different methods for calculating the Jacobian like the Pace and Barry (1997) grid evaluation of ρ in the interval $[-1, \ldots, +1]$ and spline approximations between grid points or the Chebyshev approximation of the log-determinant (Pace and LeSage, 2004). All of them are implemented in package **spdep**. For the estimation of the probit models we are using the existing facilities in **spdep** (`do_ldet()`).

Computation of marginal effects

In spatial lag models (and its probit variants), a change of an explanatory variable x_{ir} will affect both the same response variable z_i (direct effects) and possibly all other responses z_j ($j \neq i$; indirect effects or spatial spillovers). The resulting effects matrix $S_r(W)$ is $n \times n$ for $\mathbf{x}_r$ ($r = 1, \ldots, k$) and three summary measures of $S_r(W)$ will be computed: average direct effects ($M_r(D) = n^{-1} tr S_r(W)$), average total effects ($M_r(T) = n^{-1} 1_n' S_r(W) 1_n$) and the average indirect effects as the difference of total and direct effects ($M_r(I) = M_r(T) - M_r(D)$). In contrast to the SAR probit, there are no spatial spill-over effects in the SEM probit model (2). In the MATLAB spatial econometrics toolbox (LeSage, 2010), the computation of the average total effects requires an inversion of the $n \times n$ matrix $\mathbf{S} = (\mathbf{I}_n - \rho \mathbf{W})$ at every MCMC iteration. Using the same QR-decomposition of $\mathbf{S}$ as described above and solving the equation `solve(qr(S),rep(1,n))` speeds up the calculation of total effects by magnitudes.

Examples

Package spatialprobit

After describing the implementation issues to ensure that the estimation is fast enough and capable to handle large problems, we briefly describe the interface of the methods in package **spatialprobit** (Wilhelm and de Matos, 2013) and then turn to some examples.

The main estimation method for the SAR probit model `sar_probit_mcmc(y,X,W)` takes a vector of dependent variables `y`, a model matrix `X` and a spatial weight matrix `W`. The method `sarprobit(formula, W,data)` is a wrapper which allows a model formula and a data frame. Both methods require a spatial weight matrix `W` to be passed as an argument. Additionally, the number of MCMC start values, the number of burn-in iterations and a thinning parameter can be specified. The estimation `fit <-sarprobit(y ~ x,W,data)` returns an object of class `sarprobit`. The model coefficients can be extracted via `coef(fit)`. `summary(fit)` returns a coefficient table with z-values, `impacts(fit)` gives the marginal effects and the `plot(fit)` method provides MCMC trace plots, posterior density plots as well as autocorrelation plots of the model parameters. `logLik(fit)` and `AIC(fit)` return the log likelihood and the AIC of the model for model comparison and testing.

Experiment from LeSage/Pace

We replicate the experiment from LeSage and Pace (2009, section 10.1.5) for $n = 400$ and $n = 1000$ random points in a plane and a spatial weight matrix with the 6 nearest neighbors. The spatial probit model parameters are $\sigma_\epsilon^2 = 1$, $\boldsymbol{\beta} = (0, 1, -1)'$ and $\rho = 0.75$. We generate data from this model with the following code.

```
> library(spatialprobit)
> set.seed(2)
> n <- 400
> beta <- c(0, 1, -1)
> rho <- 0.75
```

```
> X <- cbind(intercept = 1, x = rnorm(n), y = rnorm(n))
> I_n <- sparseMatrix(i = 1:n, j = 1:n, x = 1)
> nb <- knn2nb(knearneigh(cbind(x = rnorm(n), y = rnorm(n)), k = 6))
> listw <- nb2listw(nb, style = "W")
> W <- as(as_dgRMatrix_listw(listw), "CsparseMatrix")
> eps <- rnorm(n = n, mean = 0, sd = 1)
> z <- solve(qr(I_n - rho * W), X %*% beta + eps)
> y <- as.double(z >= 0)
```

We estimate the spatial probit model as

```
> sarprobit.fit1 <- sarprobit(y ~ X - 1, W, ndraw = 1000, burn.in = 200,
+     thinning = 1, m = 10)
> summary(sarprobit.fit1)
> plot(sarprobit.fit1)
```

Table 2 shows the results of the SAR probit estimation for the same experiment as in LeSage and Pace (2009). The results shown there can be replicated either using the code above (for $n = 400$) or using the `LeSagePaceExperiment()` function in the package and setting `set.seed(2)` (for $n = 400$ and $n = 1000$)):

```
> set.seed(2)
> res1 <- LeSagePaceExperiment(n = 400, beta = c(0, 1, -1), rho = 0.75,
+     ndraw = 1000, burn.in = 200, thinning = 1, m = 10)
> summary(res1)
> set.seed(2)
> res2 <- LeSagePaceExperiment(n = 1000, beta = c(0, 1, -1), rho = 0.75,
+     ndraw = 1000, burn.in = 200, thinning = 1, m = 1)
> summary(res2)
```

The corresponding ML and GMM estimates in Table 2 can be replicated by creating the data as above (replacing `n <-1000` for $n = 1000$ accordingly) and

```
> library(McSpatial)
> wmat <- as.matrix(W)
> mle.fit1 <- spprobitml(y ~ X[, "x"] + X[, "y"], wmat = wmat)
> gmm.fit1 <- spprobit(y ~ X[, "x"] + X[, "y"], wmat = wmat)
```

Our Bayesian estimation yields similar results, but our implementation is much faster (factor 20-100) than the LeSage implementation, even on a single core. The **McSpatial** GMM estimation seems to be biased in this example. The evaluation of the performance relative to **McSpatial** ML clearly needs to take into account a number of factors: First, the choice of the number of MCMC iterations N and `burn.in` samples, as well as m do control the estimation time to a large degree. Second, the current ML implementation in **McSpatial** does not compute marginal effects. Finally, **McSpatial** works with dense matrices which is superior for smaller sample sizes, but will not scale for larger n.

Table 3 shows that a change of the Gibbs sampler burn-in size m for drawing $\mathbf{z}$ has little effect on the mean and the variance of the estimates. $m = 1$ means taking the previous value of $\mathbf{z}$, $m = 2, 5, 10$ start the chain from $\mathbf{z} = 0$. Clearly, choosing $m = 2$ might not be enough as burn-in phase.

Random graph example

As a second example we present the estimation of the probit model based on a random undirected graph with $n = 200$ nodes and an average branching factor of 3. Of course this estimation procedure can be applied to real network structures such as from social networks. The package **igraph** can be used to create random graphs and to compute the adjacency matrix $\mathbf{A}$ as well as the spatial weight matrix $\mathbf{W}$ (Csardi and Nepusz, 2006).

```
> library(igraph)
> library(spatialprobit)
> set.seed(1)
> n <- 200
> branch <- 3
> probability <- branch/n
> grandom <- igraph::erdos.renyi.game(n = n, p.or.m = probability,
+     type = "gnp", directed = F, loops = F)
> V(grandom)$name <- 1:n
> A <- igraph::get.adjacency(grandom, type = "both", sparse = TRUE)
```

```
> W <- A/ifelse(rowSums(A) == 0, 1, rowSums(A))
> plot(grandom, layout = layout.fruchterman.reingold, vertex.label.family = "sans",
+     vertex.size = 2, vertex.label = "")
```

Figure 1 shows the resulting random graph.

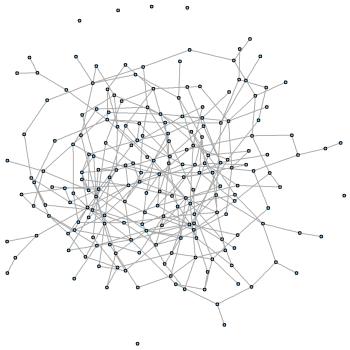


Figure 1: Random graph with $n = 200$ nodes and average branching factor 3

Next we are going to estimate the spatial probit model with $N = 3000$ draws and compare it to the standard probit model which neglects the spatial dependencies in the network.

```
> set.seed(1.2345)
> x <- rnorm(n)
> X <- cbind(const = rep(1, n), x = x)
> p <- 0.3
> beta <- c(-1, 2)
> I_n <- sparseMatrix(i = 1:n, j = 1:n, x = 1)
> z <- solve(qr(I_n - p * W), X %*% beta + rnorm(n))
> y <- as.numeric(z >= 0)
> sarprobit.fit <- sarprobit(y ~ X - 1, W, ndraw = 3000, burn.in = 200,
+                            thinning = 1)
```

The true parameter in this model are $\boldsymbol{\beta} = (-1, 2)'$ and $\rho = 0.3$.

```
> summary(sarprobit.fit)

--------MCMC spatial autoregressive probit--------
Execution time  = 25.350 secs

N draws         =   3000, N omit (burn-in)=    200
N observations  =    200, K covariates    =      2
# of 0 Y values =    151, # of 1 Y values =     49
Min rho         = -1.000, Max rho         =  1.000
---------------------------------------------------

       Estimate Std. Dev  p-level t-value Pr(>|z|)
Xconst -1.25361  0.20035  0.00000   -6.26  2.3e-09 ***
Xx      2.05238  0.28529  0.00000    7.19  1.2e-11 ***
rho     0.24796  0.10571  0.00967    2.35     0.02 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The direct, indirect and total marginal effects are extracted using

```
> impacts(sarprobit.fit)

--------Marginal Effects--------

(a) Direct effects
   lower_005 posterior_mean upper_095
Xx     0.231          0.268       0.3

(b) Indirect effects
   lower_005 posterior_mean upper_095
Xx    -0.299         -0.266     -0.23

(c) Total effects
   lower_005 posterior_mean upper_095
Xx   0.00149        0.00179         0
```

The corresponding non-spatial probit model is estimated using the `glm()` function:

```
> glm1 <- glm(y ~ x, family = binomial("probit"))
> summary(glm1, digits = 4)

Call:
glm(formula = y ~ x, family = binomial("probit"))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2337  -0.3488  -0.0870  -0.0002   2.4107

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -1.491      0.208   -7.18  7.2e-13 ***
x              1.966      0.281    6.99  2.7e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 222.71  on 199  degrees of freedom
Residual deviance: 103.48  on 198  degrees of freedom
AIC: 107.5

Number of Fisher Scoring iterations: 7
```

Figures 2 and 3 show the trace plots and posterior densities as part of the MCMC estimation results. Table 4 compares the SAR probit and standard probit estimates.

Diffusion of innovation and information

A last example looks at the information flow in social networks. Coleman et al. (1957, 1966) have studied how innovations (i.e. new drugs) diffuse among physicians, until the new drug is widely adopted by all physicians. They interviewed 246 physicians in 4 different US cities and looked at the role of 3 interpersonal networks (friends, colleagues and discussion circles) in the diffusion process (November, 1953–February, 1955). Figure 4 illustrates one of the 3 social structures based on the question "To whom do you most often turn for advice and information?". The data set is available at `http://moreno.ss.uci.edu/data.html#ckm`, but also part of **spatialprobit** (`data(CKM)`). See also Burt (1987) and den Bulte and Lilien (2001) for further discussion of this data set. The dependent variable in the model is the month in which a doctor first prescribed the new drug. Explanatory variables are the social structure and individual variables.

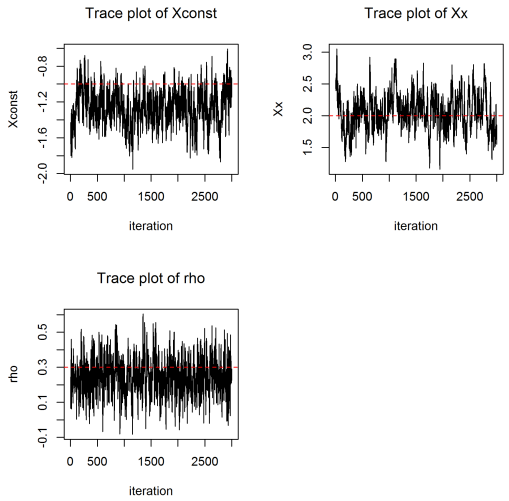


Figure 2: MCMC trace plots for model parameters with horizontal lines marking the true parameters

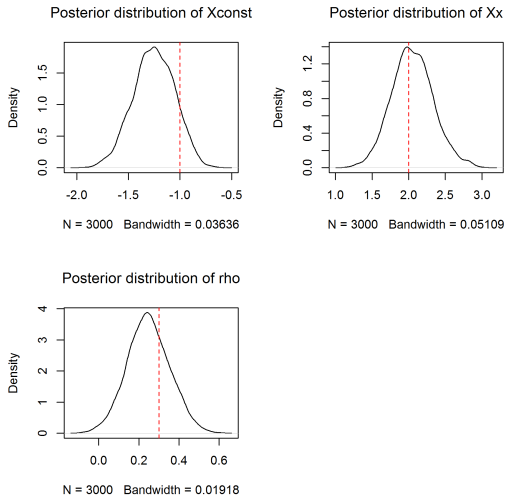


Figure 3: Posterior densities for model parameters with vertical markers for the true parameters

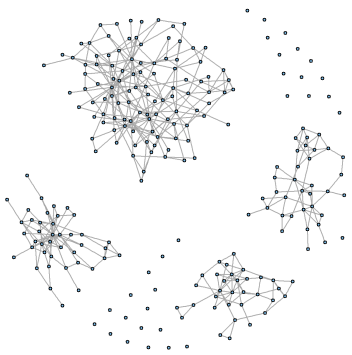


Figure 4: Social network of physicians in 4 different cities based on "advisorship" relationship

	LeSage (2009)		sarprobit		McSpatial ML		McSpatial GMM	
Estimates	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
	$n = 400$, $m = 10$				$n = 400$			
$\beta_1 = 0$	-0.1844	0.0686	0.0385	0.0562	0.0286	0.0302	-0.1042	0.0729
$\beta_2 = 1$	0.9654	0.1179	0.9824	0.1139	1.0813	0.1115	0.7690	0.0815
$\beta_3 = -1$	-0.8816	0.1142	-1.0014	0.1163	-0.9788	0.1040	-0.7544	0.0829
$\rho = 0.75$	0.6653	0.0564	0.7139	0.0427	0.7322	0.0372	1.2208	0.1424
Time (sec.)	1,276† / 623‡		11.5		1.4§		0.0§	
	$n = 1000$, $m = 1$				$n = 1000$			
$\beta_1 = 0$	0.05924	0.0438	-0.0859	0.0371	0.0010	0.0195	-0.1045	0.0421
$\beta_2 = 1$	0.96105	0.0729	0.9709	0.0709	1.0608	0.0713	0.7483	0.0494
$\beta_3 = -1$	-1.04398	0.0749	-0.9858	0.0755	-1.1014	0.0728	-0.7850	0.0528
$\rho = 0.75$	0.69476	0.0382	0.7590	0.0222	0.7227	0.0229	1.4254	0.0848
Time (sec.)	586† / 813‡		15.5		18.7§		0.0§	

Table 2: Bayesian SAR probit estimates for $n = 400$ and $n = 1000$ with $N = 1000$ draws and 200 burn-in samples. m is the burn-in size in the Gibbs sampler drawing $\mathbf{z}$. Timings in R include the computation of marginal effects and were measured using R 2.15.2 on an Intel® Core™i7-2600 CPU @3.40 GHz. Times marked with (†) are taken from LeSage and Pace (2009, Table 10.1). To allow for a better comparison, these models were estimated anew (‡) using MATLAB R2007b and the spatial econometrics toolbox (LeSage, 2010) on the very same machine used for getting the R timings. The ML and GMM timings (§) do not involve the computation of marginal effects.

	m=1		m=2		m=5		m=10	
Estimates	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
$\beta_1 = 0$	0.0385	0.0571	0.0447	0.0585	0.0422	0.0571	0.0385	0.0562
$\beta_2 = 1$	1.0051	0.1146	0.8294	0.0960	0.9261	0.1146	0.9824	0.1139
$\beta_3 = -1$	-1.0264	0.1138	-0.8417	0.0989	-0.9446	0.1138	-1.0014	0.1163
$\rho = 0.75$	0.7226	0.0411	0.6427	0.0473	0.6922	0.0411	0.7139	0.0427
Time (sec.)	10.2		10.2		10.5		11.0	
Time (sec.) in LeSage (2009)	195		314		-		1270	

Table 3: Effects of the Gibbs sampler burn-in size m on SAR probit estimates for $n = 400$, $N = 1000$ draws and `burn.in=200`

	SAR probit			Probit		
Estimates	Mean	Std dev	p-level	Mean	Std dev	p-level
$\beta_1 = -1$	-1.2536	0.2004	0.0000	-1.4905	0.2077	0.0000
$\beta_2 = 2$	2.0524	0.2853	0.0000	1.9656	0.2812	0.0000
$\rho = 0.3$	0.2480	0.1057	0.0097			
Time (sec.)	25.4					

Table 4: SAR probit estimates vs. probit estimates for the random graph example

	Probit		SAR Probit	
Estimates	Mean	z value	Mean	z value
(Intercept)	−0.3659	−0.40	0.1309	0.16
influence	0.9042	2.67		
city	−0.0277	−0.26	−0.0371	−0.39
med_sch_yr	0.2937	2.63	0.3226	3.22
meetings	−0.1797	−1.55	−0.1813	−1.64
jours	0.1592	2.46	0.1368	2.08
free_time	0.3179	2.31	0.3253	2.58
discuss	−0.0010	−0.01	−0.0643	−0.57
clubs	−0.2366	−2.36	−0.2252	−2.45
friends	−0.0003	−0.00	−0.0185	−0.31
community	0.1974	1.65	0.2298	2.08
patients	−0.0402	−0.67	−0.0413	−0.72
proximity	−0.0284	−0.45	−0.0232	−0.40
specialty	−0.9693	−8.53	−1.0051	−8.11
ρ			0.1138	1.33

Table 5: SAR probit and standard probit estimates for the Coleman data.

We are estimating a standard probit model as well as a SAR probit model based on the "advisorship" network and trying to find determinants for all adopters of the new drug. We do not aim to fully reanalyze the Coleman data set, nor to provide a detailed discussion of the results. We rather want to illustrate the model estimation in R. We find a positive relationship for the social influence variable in the probit model, but social contagion effects as captured by $\rho\mathbf{W}$ in the more sound SAR probit model is rather small and insignificant. This result suggests that social influence is a factor in information diffusion, but the information flow might not be correctly described by a SAR model. Other drivers for adoption which are ignored here, such as marketing efforts or aggressive pricing of the new drug may play a role in the diffusion process too (den Bulte and Lilien, 2001).

```
> set.seed(12345)
> # load data set "CKM" and spatial weight matrices "W1","W2","W3"
> data(CKM)
> # 0/1 variable for early adopter
> y <- as.numeric(CKM$adoption.date <= "February, 1955")
> # create social influence variable
> influence <- as.double(W1 %*% as.numeric(y))
> # Estimate Standard probit model
> glm.W1 <- glm(y ~ influence + city + med_sch_yr + meetings + jours + free_time +
+     discuss + clubs + friends + community + patients + proximity + specialty,
+     data = CKM, family = binomial("probit"))
> summary(glm.W1, digits = 3)
> # Estimate SAR probit model without influence variable
> sarprobit.fit.W1 <- sarprobit(y ~ 1 + city + med_sch_yr + meetings + jours +
+     free_time + discuss + clubs + friends + community + patients + proximity +
+     specialty, data = CKM, W = W1)
> summary(sarprobit.fit.W1, digits = 3)
```

Table 5 presents the estimation results for the non-spatial probit and the SAR probit specification. The coefficient estimates are similar in magnitude and sign, but the estimate for ρ does not support the idea of spatial correlation in this data set.

Parallel estimation of models

MCMC is, similar to the bootstrap, an embarrassingly parallel problem. It can be easily run in parallel on several cores. From version 2.14.0, R offers a unified way of doing parallelization with the **parallel** package. There are several different approaches available to achieve parallelization and not all approaches are available for all platforms. See for example the conceptual differences between the two main methods `mclapply` and `parLapply`, where the first will only work serially on Windows. Users are therefore encouraged to read the **parallel** package documentation for choosing the appropriate way.

Here, we only sketch how easy the SAR probit estimation can be done in parallel with 2 tasks:

```
> library(parallel)
> mc <- 2
> run1 <- function(...) sarprobit(y ~ X - 1, W, ndraw = 500, burn.in = 200,
+     thinning = 1)
> system.time({
+     set.seed(123, "L'Ecuyer")
+     sarprobit.res <- do.call(c, mclapply(seq_len(mc), run1))
+ })
> summary(sarprobit.res)
```

Due to the overhead in setting up the cluster, it is reasonable to expect another 50% performance gain when working with 2 CPUs.

Summary

In this article we presented the estimation of spatial probit models in R and pointed to the critical implementation issues. Our performance studies showed that even large problems with $n = 10,000$ or $n = 100,000$ observations can be handled within reasonable time. We provided an update of **tmvtnorm** and a new package **spatialprobit** on CRAN with the methods for estimating spatial probit models implemented (Wilhelm and de Matos, 2013). The package currently implements three limited dependent models: the spatial lag probit model (`sarprobit()`), the probit model with spatial errors (`semprobit()`) and the SAR Tobit model (`sartobit()`). The Bayesian approach can be further extended to other limited dependent spatial models, such as ordered probit or models with multiple spatial weights matrices. We are planning to include these in the package in near future.

Bibliography

J. Albert. *Bayesian Computation in R*. Springer, 2007. [p131]

J. Albert. *LearnBayes: Functions for Learning Bayesian Inference*, 2012. URL http://CRAN.R-project.org/package=LearnBayes. R package version 2.12. [p131]

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2013. URL http://CRAN.R-project.org/package=Matrix. R package version 1.0-12. [p130]

R. Bivand. Computing the Jacobian in spatial models: an applied survey. Discussion paper / Norwegian School of Economics and Business Administration, Department of Economics ; 2010,20, August 2010. URL http://brage.bibsys.no/nhh/bitstream/URN:NBN:no-bibsys_brage_23930/1/dp2010-20.pdf. [p134]

R. Bivand. *spdep: Spatial dependence: weighting schemes, statistics and models*, 2013. URL http://CRAN.R-project.org/package=spdep. R package version 0.5-57. [p130]

R. S. Burt. Social contagion and innovation: Cohesion versus structural equivalence. *American Journal of Sociology*, 92(6):1287–1335, 1987. [p137]

C. T. Butts. *sna: Tools for Social Network Analysis*, 2013. URL http://CRAN.R-project.org/package=sna. R package version 2.3-1. [p130]

C. T. Butts, M. S. Handcock, and D. R. Hunter. *network: Classes for Relational Data*. Irvine, CA, 2013. URL http://statnet.org/. R package version 1.7-2. [p130]

J. Coleman, E. Katz, and H. Menzel. The diffusion of an innovation among physicians. *Sociometry*, 20: 253–270, 1957. [p137]

J. S. Coleman, E. Katz, and H. Menzel. *Medical Innovation: A Diffusion Study*. New York: Bobbs Merrill, 1966. [p137]

G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL http://igraph.sf.net. [p135]

C. V. den Bulte and G. L. Lilien. Medical innovation revisited: Social contagion versus marketing effort. *American Journal of Sociology*, 106(5):1409–1435, 2001. [p137, 140]

P. Diggle and P. Ribeiro. *Model Based Geostatistics*. Springer, New York, 2007. [p130]

A. O. Finley and S. Banerjee. *spBayes: Univariate and Multivariate Spatial Modeling*, 2013. URL http://CRAN.R-project.org/package=spBayes. R package version 0.3-7. [p130]

R. J. Franzese, J. C. Hays, and S. Cook. Spatial-, temporal-, and spatiotemporal-autoregressive probit models of interdependent binary outcomes: Estimation and interpretation. Prepared for the Spatial Models of Politics in Europe & Beyond, Texas A&M University, February 2013. [p131]

A. Genz, F. Bretz, T. Miwa, X. Mi, F. Leisch, F. Scheipl, and T. Hothorn. *mvtnorm: Multivariate Normal and t Distributions*, 2013. URL http://CRAN.R-project.org/package=mvtnorm. R package version 0.9-9995. [p131]

J. F. Geweke. Efficient simulation from the multivariate normal and Student-t distributions subject to linear constraints and the evaluation of constraint probabilities, 1991. URL http://www.biz.uiowa.edu/faculty/jgeweke/papers/paper47/paper47.pdf. [p132]

J. F. Geweke. *Contemporary Bayesian Econometrics and Statistics*. John Wiley and Sons, 2005. [p132]

T. Klier and D. P. McMillen. Clustering of auto supplier plants in the United States: Generalized method of moments spatial probit for large samples. *Journal of Business and Economic Statistics*, 26: 460–471, 2008. [p131]

R. Koenker and P. Ng. *SparseM: Sparse Linear Algebra*, 2013. URL http://CRAN.R-project.org/package=SparseM. R package version 0.99. [p130]

J. LeSage and R. K. Pace. *Introduction to Spatial Econometrics*. CRC Press, 2009. [p130, 131, 132, 133, 134, 135, 139]

J. P. LeSage. Bayesian estimation of limited dependent variable spatial autoregressive models. *Geographical Analysis*, 32(1):19–35, 2000. [p130]

J. P. LeSage. Spatial econometrics toolbox for MATLAB, March 2010. URL http://www.spatial-econometrics.com/. [p134, 139]

J. P. LeSage, R. K. Pace, N. Lam, R. Campanella, and X. Liu. New Orleans business recovery in the aftermath of hurricane Katrina. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 174:1007–1027, 2011. [p131]

J. J. Majure and A. Gebhardt. *sgeostat: An Object-oriented Framework for Geostatistical Modeling in S+*, 2013. URL http://CRAN.R-project.org/package=sgeostat. R package version 1.0-25; S original by James J. Majure Iowa State University and R port + extensions by Albrecht Gebhardt. [p130]

T. L. Marsh, R. C. Mittelhammer, and R. G. Huffaker. Probit with spatial correlation by field plot: Potato leafroll virus net necrosis in potatoes. *Journal of Agricultural, Biological, and Environmental Statistics*, 5:22–36, 2000. URL http://www.jstor.org/stable/1400629. [p131]

D. McMillen. *McSpatial: Nonparametric spatial data analysis*, 2013. URL http://CRAN.R-project.org/package=McSpatial. R package version 2.0. [p130, 131]

R. K. Pace and R. Barry. Quick computation of spatial autoregressive estimators. *Geographical Analysis*, 29:232–247, 1997. [p134]

R. K. Pace and J. LeSage. Chebyshev approximation of log-determinants of spatial weight matrices. *Computational Statistics & Data Analysis*, 45(2):179–196, 2004. [p134]

G. Piras. sphet: Spatial models with heteroskedastic innovations in R. *Journal of Statistical Software*, 35 (1):1–21, 2010. URL http://www.jstatsoft.org/v35/i01/. [p130]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, 2002. [p130]

S. Wilhelm and M. G. de Matos. *spatialprobit: Spatial Probit Models*, 2013. URL http://CRAN.R-project.org/package=spatialprobit. R package version 0.9-9. [p130, 134, 141]

S. Wilhelm and B. G. Manjunath. tmvtnorm: A package for the truncated multivariate normal distribution. *The R Journal*, 2(1):25–29, June 2010. URL http://journal.r-project.org/archive/2010-1/RJournal_2010-1_Wilhelm+Manjunath.pdf. [p132]

S. Wilhelm and B. G. Manjunath. *tmvtnorm: Truncated Multivariate Normal and Student t Distribution*, 2013. URL http://CRAN.R-project.org/package=tmvtnorm. R package version 1.4-8. [p131, 132]

J. M. Wooldridge. *Introductory Econometrics - A Modern Approach*. South Western College Publishing, 2002. [p131]

Stefan Wilhelm
Department of Finance, WWZ (Wirtschaftswissenschaftliches Zentrum)
University of Basel
Switzerland
wilhelm@financial.com

Miguel Godinho de Matos
Department of Engineering & Public Policy
Carnegie Mellon University
United States
miguelgodinhomatos@cmu.edu

ggmap: Spatial Visualization with ggplot2

by David Kahle and Hadley Wickham

Abstract In spatial statistics the ability to visualize data and models superimposed with their basic social landmarks and geographic context is invaluable. **ggmap** is a new tool which enables such visualization by combining the spatial information of static maps from Google Maps, OpenStreetMap, Stamen Maps or CloudMade Maps with the layered grammar of graphics implementation of **ggplot2**. In addition, several new utility functions are introduced which allow the user to access the Google Geocoding, Distance Matrix, and Directions APIs. The result is an easy, consistent and modular framework for spatial graphics with several convenient tools for spatial data analysis.

Introduction

Visualizing spatial data in R can be a challenging task. Fortunately the task is made a good deal easier by the data structures and plot methods of **sp**, **RgoogleMaps**, and related packages (Pebesma and Bivand, 2006; Bivand et al., 2008; Loecher and Berlin School of Economics and Law, 2013). Using those methods, one can plot the basic geographic information of (for instance) a shape file containing polygons for areal data or points for point referenced data. However, compared to specialized geographic information systems (GISs) such as ESRI's ArcGIS, which can plot points, polygons, etc. on top of maps and satellite imagery with drag-down menus, these visualizations can be pretty disappointing. This article details some new methods for the visualization of spatial data in R using the layered grammar of graphics implementation of **ggplot2** in conjunction with the contextual information of static maps from Google Maps, OpenStreetMap, Stamen Maps or CloudMade Maps (Wickham, 2009, 2010). The result is an easy to use R package named **ggmap**. After describing the nuts and bolts of **ggmap**, we showcase some of its capabilities in a simple case study concerning violent crimes in downtown Houston, Texas and present an overview of a few utility functions.

Plotting spatial data in R

Areal data is data which corresponds to geographical extents with polygonal boundaries. A typical example is the number of residents per zip code. Considering only the boundaries of the areal units, we are used to seeing areal plots in R which resemble those in Figure 1 (left).

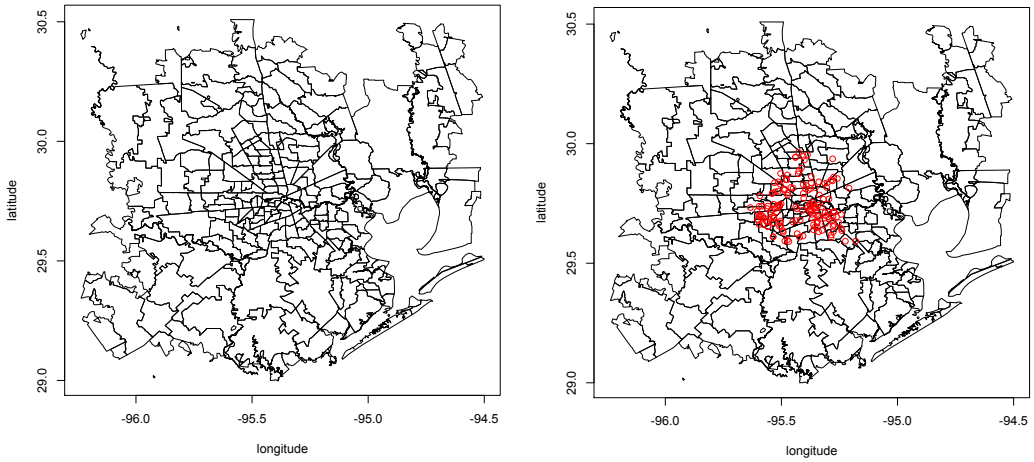


Figure 1: A typical R areal plot – zip codes in the Greater Houston area (left), and a typical R spatial scatterplot – murders in Houston from January 2010 to August 2010 (right).

While these kinds of plots are useful, they are not as informative as we would like in many situations. For instance, when plotting zip codes it is helpful to also see major roads and other landmarks which form the boundaries of areal units.

The situation for point referenced spatial data is often much worse. Since we can't easily contextualize a scatterplot of points without any background information at all, it is common to add points as

an overlay of some areal data—whatever areal data is available. The resulting plot looks like Figure 1 (right).

In most cases the plot is understandable to the researcher who has worked on the problem for some time but is of hardly any use to his audience, who must work to associate the data of interest with their location. Moreover, it leaves out many practical details—are most of the events to the east or west of landmark x? Are they clustered around more well-to-do parts of town, or do they tend to occur in disadvantaged areas? Questions like these can't really be answered using these kinds of graphics because we don't think in terms of small scale areal boundaries (e.g. zip codes or census tracts).

With a little effort better plots can be made, and tools such as **maps**, **maptools**, **sp**, or **RgoogleMaps** make the process much easier; in fact, **RgoogleMaps** was the inspiration for **ggmap** (Becker et al., 2013; Bivand and Lewin-Koh, 2013).

Moreover, there has recently been a deluge of interest in the subject of mapmaking in R—Ian Fellows' excellent interactive GUI-driven **DeducerSpatial** package based on Bing Maps comes to mind (Fellows et al., 2013). **ggmap** takes another step in this direction by situating the contextual information of various kinds of static maps in the **ggplot2** plotting framework. The result is an easy, consistent way of specifying plots which are readily interpretable by both expert and audience and safeguarded from graphical inconsistencies by the layered grammar of graphics framework. The result is a spatial plot resembling Figure 2. Note that map images and information in this work may appear slightly different due to map provider changes over time.

```
murder <- subset(crime, offense == "murder")
qmplot(lon, lat, data = murder, colour = I('red'), size = I(3), darken = .3)
```

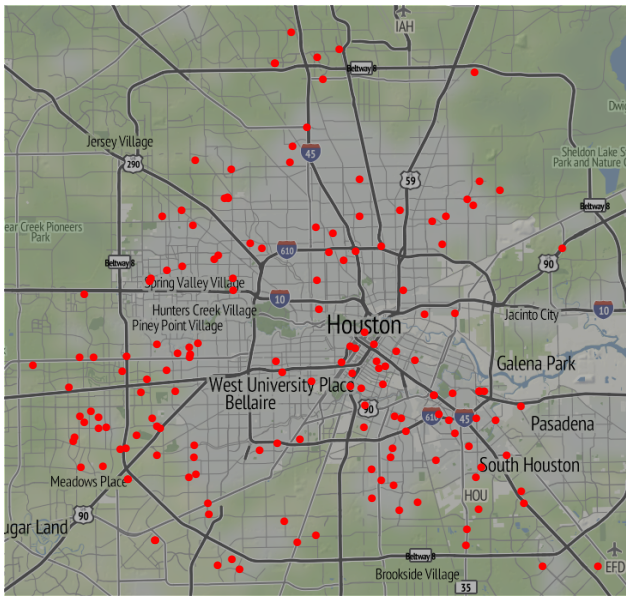


Figure 2: A spatial scatterplot based on Stamen Maps' terrain tile set made with the `qmplot` function, an experimental amalgamation of the functions presented in this article.

The layered grammar of graphics

One advantage of making the plots with **ggplot2** is the layered grammar of graphics on which **ggplot2** is based (Wickham, 2010; Wilkinson, 2005). By definition, the layered grammar demands that every plot consist of five components :

- a default dataset with aesthetic mappings,
- one or more layers, each with a geometric object ("geom"), a statistical transformation ("stat"), and a dataset with aesthetic mappings (possibly defaulted),
- a scale for each aesthetic mapping (which can be automatically generated),
- a coordinate system, and
- a facet specification.

Since **ggplot2** is an implementation of the layered grammar of graphics, every plot made with **ggplot2** has each of the above elements. Consequently, **ggmap** plots also have these elements, but certain elements are fixed to map components : the x aesthetic is fixed to longitude, the y aesthetic is fixed to latitude, and the coordinate system is fixed to the Mercator projection.[1]

The major theoretical advantage of using the layered grammar in plotting maps is that aesthetic scales are kept consistent. In the typical situation where the map covers the extent of the data, in **ggmap** the latitude and longitude scales key off the map (by default) and one scale is used for those axes. The same is true of colors, fills, alpha blendings, and other aesthetics which are built on top of the map when other layers are presented—each is allotted one scale which is kept consistent across each layer of the plot. This aspect of the grammar is particularly important for faceted plots in order to make a proper comparison across several plots. Of course, the scales can still be tricked if the user improperly specifies the spatial data, e.g. using more than one projection in the same map, but fixing such errors is beyond any framework.

The practical advantage of using the grammar is even better. Since the graphics are done in **ggplot2** the user can draw from the full range of **ggplot2**'s capabilities to layer elegant visual content—geoms, stats, scales, etc.—using the usual **ggplot2** coding conventions. This was already seen briefly in Figure 2 where the arguments of `qmplot` are identical to that of **ggplot2**'s `qplot`; much more will be seen shortly.

How ggmap works

The basic idea driving **ggmap** is to take a downloaded map image, plot it as a context layer using **ggplot2**, and then plot additional content layers of data, statistics, or models on top of the map. In **ggmap** this process is broken into two pieces – (1) downloading the images and formatting them for plotting, done with `get_map`, and (2) making the plot, done with `ggmap`. `qmap` marries these two functions for quick map plotting (c.f. **ggplot2**'s `ggplot`), and `qmplot` attempts to wrap up the entire plotting process into one simple command (c.f. **ggplot2**'s `qplot`).

The `get_map` function

In **ggmap**, downloading a map as an image and formatting the image for plotting is done with the `get_map` function. More specifically, `get_map` is a wrapper function for the underlying functions `get_googlemap`, `get_openstreetmap`, `get_stamenmap`, and `get_cloudmademap` which accepts a wide array of arguments and returns a classed raster object for plotting with `ggmap`.

As the most important characteristic of any map is location, the most important argument of `get_map` is the `location` argument. Ideally, `location` is a longitude/latitude pair specifying the center of the map and accompanied by a `zoom` argument, an integer from 3 to 20 specifying how large the spatial extent should be around the center, with 3 being the continent level and 20 being roughly the single building level. `location` is defaulted to downtown Houston, Texas, and `zoom` to 10, roughly a city-scale.

While longitude/latitude pairs are ideal for specifying a location, they are somewhat inconvenient on a practical level. For this reason, `location` also accepts a character string. The string, whether containing an address, zip code, or proper name, is then passed to the `geocode` function which then determines the appropriate longitude/latitude coordinate for the center. In other words, there is no need to know the exact longitude/latitude coordinates of the center of the map—`get_map` can determine them from more colloquial ("lazy") specifications so that they can be specified very loosely. For example, since

```
> geocode("the white house")
      lon       lat
-77.03676  38.89784
```

works, `"the white house"` is a viable `location` argument. More details on `geocode` and other utility functions are discussed at the end of this article.

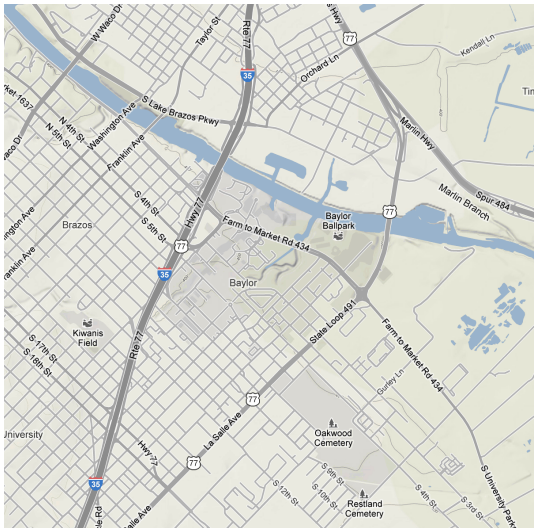
In lieu of a center/zoom specification, some users find a bounding box specification more convenient. To accommodate this form of specification, `location` also accepts numeric vectors of length four following the left/bottom/right/top convention. This option is not currently available for Google Maps.

While each map source has its own web application programming interface (API), specification of `location`/`zoom` in `get_map` works for each by computing the appropriate parameters (if necessary)

[1] Note that because of the Mercator projection limitations in `mapproject`, anything above/below $\pm 80^\circ$ cannot be plotted currently.

and passing them to each of the API specific `get_*` functions. To ensure that the resulting maps are the same across the various sources for the same `location/zoom` specification, `get_map` first grabs the appropriate Google Map, determines its bounding box, and then downloads the other map as needed. In the case of Stamen Maps and CloudMade Maps, this involves a stitching process of combining several tiles (small map images) and then cropping the result to the appropriate bounding box. The result is a single, consistent specification syntax across the four map sources as seen for Google Maps and OpenStreetMap in Figure 3.

```
baylor <- "baylor university"
qmap(baylor, zoom = 14)
```




```
qmap(baylor, zoom = 14, source = "osm")
```

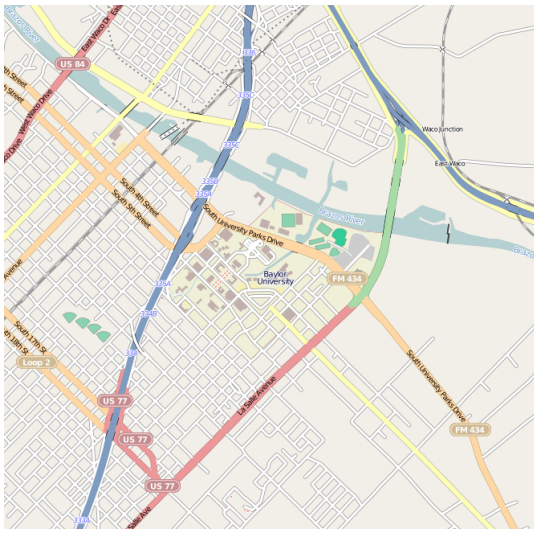


Figure 3: `get_map` provides the same spatial extent for Google Maps (top) and OpenStreetMaps (bottom) with a single simple syntax, even though their APIs are quite different.

Before moving into the `source` and `maptype` arguments, it is important to note that the underlying API specific `get_*` functions for which `get_map` is a wrapper provide more extensive mechanisms for downloading from their respective sources. For example, `get_googlemap` can access almost the full range of the Google Static Maps API as seen in Figure 4.

Tile style – the `source` and `maptype` arguments of `get_map`

The most attractive aspect of using different map sources (Google Maps, OpenStreetMap, Stamen Maps, and CloudMade Maps) is the different map styles provided by the producer. These are specified

```
set.seed(500)
df <- round(data.frame(
  x = jitter(rep(-95.36, 50), amount = .3),
  y = jitter(rep( 29.76, 50), amount = .3)
), digits = 2)

map <- get_googlemap('houston', markers = df, path = df, scale = 2)

ggmap(map, extent = 'device')
```

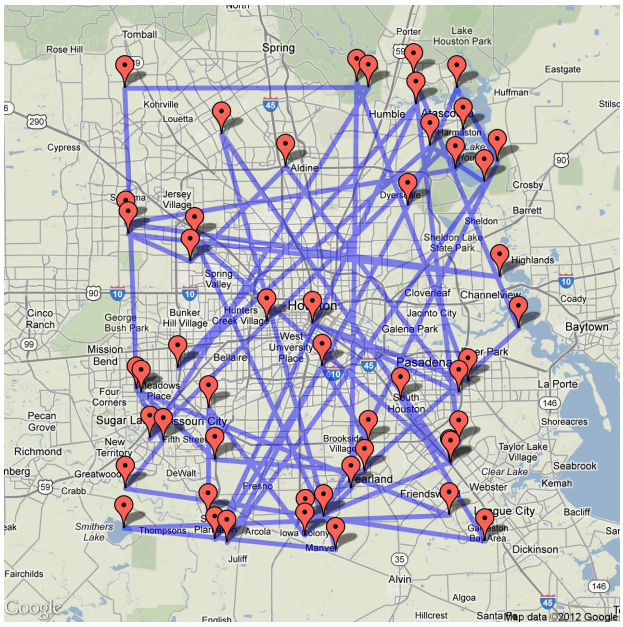


Figure 4: Accessing Google Maps API features with `get_googlemap`.

with the `maptype` argument of `get_map` and must agree with the `source` argument. Some styles emphasize large roadways, others bodies of water, and still others political boundaries. Some are better for plotting in a black-and-white medium; others are simply nice to look at. This section gives a run down of the various map styles available in **ggmap**.

Google provides four different familiar types—terrain (default), satellite (e.g. Figure 13), roadmap, and hybrid (e.g. Figure 12). OpenStreetMap, on the other hand, only provides the default style shown in Figure 3.

Style is where Stamen Maps and CloudMade Maps really shine. Stamen Maps has three available tile sets—terrain (e.g. Figures 2 or 13), watercolor, and toner (for the latter two see Figure 5).

```
qmap(baylor, zoom = 14, source = "stamen", maptype = "watercolor")
qmap(baylor, zoom = 14, source = "stamen", maptype = "toner")
```

Figure 5: Stamen tile sets `maptype = "watercolor"` and `maptype = "toner"`.

Stamen's terrain tile set is quite similar to Google's, but obviously the watercolor and toner tile sets are substantially different than any of the four Google tile sets. The latter, for example, is ideal for black-and-white plotting.

CloudMade Maps takes the tile styling even further by allowing the user to either (1) select among *thousands* of user-made sets or (2) create an entirely new style with a simple online editor where the user can specify colors, lines, and so forth for various types of roads, waterways, landmarks, etc., all of which are generated by CloudMade and accessible in **ggmap**. **ggmap**, through `get_map` (or `get_cloudmademap`) allows for both options. This is a unique feature of CloudMade Maps which really boosts their applicability and expands the possibilities with **ggmap**. The one minor drawback to using CloudMade Maps is that the user must register with CloudMade to obtain an API key and then pass the API key into `get_map` with the `api_key` argument. API keys are free of charge and can be obtained in a matter of minutes. Two low-light CloudMade map styles are seen in Figure 6. Note that map styles are only available to the user that owns them.

Both Stamen Maps and CloudMade Maps are built using OpenStreetMap data. These data are contributed by an open community of online users in much the same way Wikipedia is—both are free, both are user-contributed, and both are easily edited. Moreover, OpenStreetMap has data not only on roadways and bodies of water but also individual buildings, fountains, stop signs and other apparent minutiae. The drawback is that (like Google Maps) not all locations are mapped with the same degree of precision, and imperfections may be observed in small-scale out of the way features.[2]

The `ggmap` function

Once `get_map` has grabbed the map of interest, `ggmap` is ready to plot it. The result of `get_map` is a specially classed raster object (a matrix of colors as hexadecimal character strings) –

```
> paris <- get_map(location = "paris")
> str(paris)
```

[2] As an example, the reader is referred to look at Google Maps satellite images of northwest tributaries to Lake Waco and search for them in the Stamen watercolor tile set.

```
qmap(baylor, zoom = 14, maptype = 53428, api_key = api_key,
     source = "cloudmade")
qmap("houston", zoom = 10, maptype = 58916, api_key = api_key,
     source = "cloudmade")
```

Figure 6: Two out of thousands of user made CloudMade Maps styles. The left is comparable to Figures 3 and 5, and the right contains the bodies of water in Figure 4.

```
 chr [1:1280, 1:1280] "#C6DAB6" "#C2D6B3" "#C2D6B3" ...
 - attr(*, "class")= chr [1:2] "ggmap" "raster"
 - attr(*, "bb")='data.frame':        1 obs. of  4 variables:
  ..$ ll.lat: num 48.6
  ..$ ll.lon: num 1.91
  ..$ ur.lat: num 49.1
  ..$ ur.lon: num 2.79
```

The purpose of ggmap is to take the map from the raster object to the screen, and it fulfills this purpose by creating a ggplot object which, when printed, draws the desired map in the graphics device. This is illustrated in Figure 7.

While ggmap requires a ggmap object, it accepts a handful of other arguments as well—extent, base_layer, maprange, legend, padding, and darken. With none of these additional arguments, ggmap effectively returns the following ggplot object

```
ggplot(aes(x = lon, y = lat), data = fourCorners) +
  geom_blank() + coord_map("mercator") +
  annotation_raster(ggmap, xmin, xmax, ymin, ymax)
```

where fourCorners is the data frame resulting from applying expand.grid to the longitude and latitude ranges specified in the bb attribute of the ggmap object. Thus, the default base layer of the ggplot2 object created by ggmap is ggplot(aes(x = lon,y = lat),data = fourCorners), and the default x and y aesthetic scales are calculated based on the longitude and latitude ranges of the map.

The extent argument dictates how much of the graphics device is covered by the map. It accepts three possible strings: "normal" shown in Figure 7, "panel" shown in Figures 10 and 12, and "device" shown in every other figure. "normal" situates the map with the usual axis padding provided by **ggplot2** and, consequently, one can see the panel behind it. "panel" eliminates this, setting the limits of the plot panel to be the longitude and latitude extents of the map with scale_[x,y]_continuous(expand = c(0,0)). "device" takes this to the extreme by eliminating the axes themselves with the new exported theme_nothing.

base_layer is a call which substitutes the default base layer to the user's specification. Thus, in the above code the user can change ggplot(aes(x = lon,y = lat),data = fourCorners) to a different call. This is essential for faceting plots since the referent of **ggplot2** functions facet_wrap and facet_grid is the base layer. Since changing the base layer changes the base scales and therefore limits of the plot, it is possible that when the base layer is changed only part of the map is visible. Setting the maprange argument to TRUE (it defaults to FALSE) ensures that *the map* determines the x and y axis

```
ggmap(paris, extent = "normal")
```

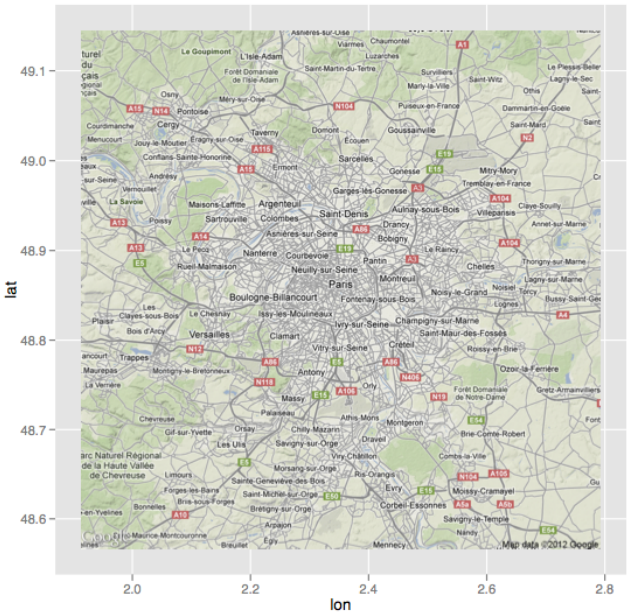


Figure 7: Setting `extent = "normal"` in `ggmap` illustrates how maps in **ggmap** are simply **ggplot2** graphics.

limits (longitude and latitude) via the `bb` attribute of the ggmap object itself, and not the `base_layer` argument.

The legend-related arguments of `ggmap` are `legend` and `padding`, and they are only applicable when `extent = "device"`. The `legend` argument determines where a legend should be drawn on the map if one should be drawn. Its options are `"left"`, `"right"` (default), `"bottom"`, `"top"`, `"topleft"`, `"bottomleft"`, `"topright"`, `"bottomright"` and `"none"`. The first four draw the legend according to **ggplot2**'s normal specifications (without any axes); the next four draw the legend on top of the map itself similar to ArcGIS; and the last eliminates the legend altogether. `padding` governs how far from the corner the legend should be drawn.

The `darken` argument, a suggestion by Jean-Olivier Irisson, tints the map image. The default, `c(0,"black")`, indicates a fully translucent black layer, i.e. no tint at all. Generally, the first argument corresponds to an alpha blending (0 = invisible, 1 = opaque) and the second argument the color of the tint. If only a number is supplied to the `darken` argument `ggmap` assumes a black tint. The tint itself is made by adding a `geom_rect` layer on top of the map. An example is provided by Figure 2, where a black tint was added to the map to enhance the visibility of the points.

Since `ggmap` returns a ggplot object, the product of `ggmap` can itself act as a base layer in the **ggplot2** framework. This is an incredibly important realization which allows for the full range of **ggplot2** capabilities. We now illustrate many of the ways in which this can be done effectively through a case study of violent crime in downtown Houston, Texas.

ggmap in action

Data

Crime data were compiled from the Houston Police Department's website over the period of January 2010–August 2010. The data were lightly cleaned and aggregated using **plyr** (Wickham, 2011) and geocoded using Google Maps (to the center of the block, e.g., 6150 Main St.); the full data set is available in **ggmap** as the data set `crime`.

```
> str(crime)
'data.frame':        86314 obs. of  17 variables:
 $ time     : POSIXt, format: "2010-01-01 0...
 $ date     : chr  "1/1/2010" "1/1/2010" "1...
```

```
 $ hour    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ premise : chr  "18A" "13R" "20R" "20R" ...
 $ offense : chr  "murder" "robbery" "aggr...
 $ beat    : chr  "15E30" "13D10" "16E20" ...
 $ block   : chr  "9600-9699" "4700-4799" ...
 $ street  : chr  "marlive" "telephone" "w...
 $ type    : chr  "ln" "rd" "ln" "st" ...
 $ suffix  : chr  "-" "-" "-" "-" ...
 $ number  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ month   : Factor w/ 12 levels "january"...
 $ day     : Factor w/ 7 levels "monday"  ...
 $ location: chr  "apartment parking lot" ...
 $ address : chr  "9650 marlive ln" "4750 ...
 $ lon     : num  -95.4 -95.3 -95.5 -95.4 ...
 $ lat     : num  29.7 29.7 29.6 29.8 29.7...
```

Since we are only interested in violent crimes which take place downtown, we restrict the data set to those qualifiers. To determine a bounding box, we first use `gglocator`, a **ggplot2** analogue of base's `locator` function exported from **ggmap**. `gglocator` works not only for **ggmap** plots, but **ggplot2** graphics in general.

```
> # find a reasonable spatial extent
> qmap('houston', zoom = 13)
> gglocator(2)
>         lon      lat
> 1 -95.39681 29.78400
> 2 -95.34188 29.73631
>
> # only violent crimes
> violent_crimes <- subset(crime,
+   offense != "auto theft" & offense != "theft" & offense != "burglary")
>
> # order violent crimes
> violent_crimes$offense <- factor(violent_crimes$offense,
+   levels = c("robbery", "aggravated assault", "rape", "murder"))
>
> # restrict to downtown
> violent_crimes <- subset(violent_crimes,
+   -95.39681 <= lon & lon <= -95.34188 &
+    29.73631 <= lat & lat <=  29.78400)
```

The analysis performed only concerns data on the violent crimes of aggravated assault, robbery, rape and murder. Note that while some effort was made to ensure the quality of the data, the data were only leisurely cleaned and the data set may still contain errors.

Analysis

The first step we might want to take is to look at where the individual crimes took place. Modulo some simple **ggplot2** styling changes (primarily in the fonts and key-styles of the legends via **ggplot2**'s `guide` function), Figure 8 contains the code to produce the spatial bubble chart shown on the left.

One of the problems with the bubble chart is overplotting and point size—we can't really get a feel for what crimes are taking place and where. One way around this is to bin the points and drop the bins which don't have any samples in them. The result (Figure 8 right) shows us where the crimes are happening at the expense of knowing their frequency.

The binned plot is the first time we really begin to see the power of having the maps in the **ggplot2** framework. While it is actually not a very good plot, it illustrates the practical advantage of the **ggplot2** framework with the contextual information of the map—the process of splitting the data frame `violent_crimes` into chunks based on the offense variable, binning the points of each, and aggregating back into one data set to plot is all done entirely behind the scenes by **ggplot2**.

What about violent crimes in general? If we neglect the type of offense, we can get a good idea of the spatial distribution of violent crimes by using a contour plot. Since the map image itself is based on **ggplot2**'s `annotation_raster`, which doesn't have a fill aesthetic, we can access the fill aesthetic to make a filled contour plot. This is seen in Figure 9 (left).

```
theme_set(theme_bw(16))
HoustonMap <- qmap("houston", zoom = 14, color = "bw", legend = "topleft")

HoustonMap +
   geom_point(aes(x = lon, y = lat, colour = offense, size = offense),
              data = violent_crimes)

HoustonMap +
  stat_bin2d(
    aes(x = lon, y = lat, colour = offense, fill = offense),
    size = .5, bins = 30, alpha = 1/2,
    data = violent_crimes
  )
```

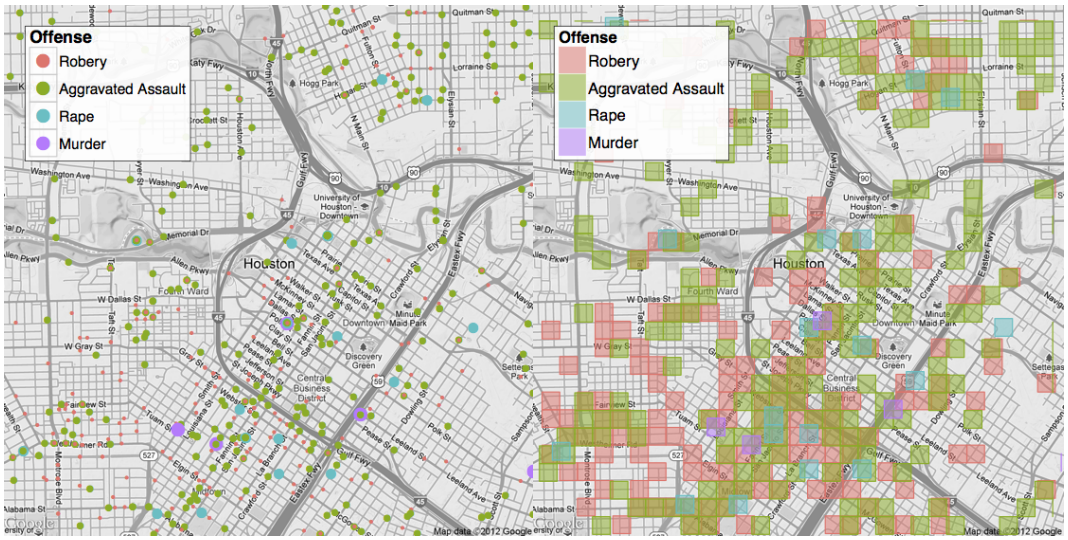


Figure 8: Violent crime bubble chart of downtown Houston (left) and the same binned (right).

```
houston <- get_map("houston", zoom = 14)
HoustonMap <- ggmap("houston", extent = "device", legend = "topleft")

HoustonMap +
  stat_density2d(
    aes(x = lon, y = lat, fill = ..level..,  alpha = ..level..),
    size = 2, bins = 4, data = violent_crimes,
    geom = "polygon"
  )

overlay <- stat_density2d(
  aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
  bins = 4, geom = "polygon",
  data = violent_crimes
)

HoustonMap + overlay + inset(
    grob = ggplotGrob(ggplot() + overlay + theme_inset()),
    xmin = -95.35836, xmax = Inf, ymin = -Inf, ymax = 29.75062
  )
```

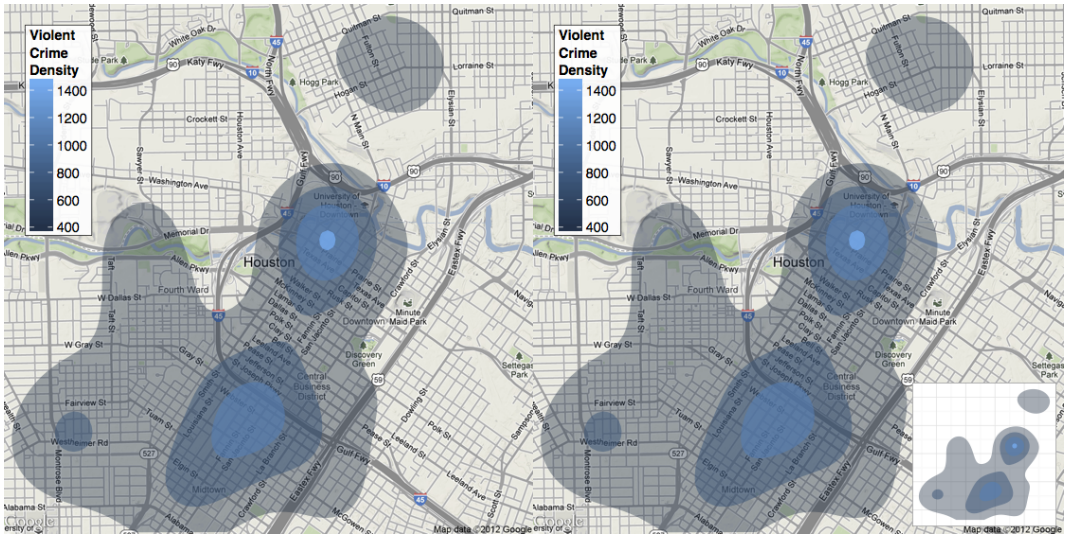


Figure 9: Filled contour plot of violent crimes (left), and the same with an inset (right).

These kinds of overlays can be incredibly effective; however, their ability to communicate information can be hindered by the fact that the map overlay can be visually confused with the map itself. This is particularly common when using colored maps. To get around this problem the `inset` function can be used to insert map insets containing the overlay on a white background with major and minor axes lines for visual guides made possible by the exported function `theme_inset`; this is seen in Figure 9 (right).

The image indicates that there are three main hotspots of activity. Each of these three corresponds to locations commonly held by Houstonians to be particularly dangerous locations. From east to west, the hotspots are caused by (1) a county jail which releases its inmates twice daily, who tend to loiter in the area indicated, (2) a commercial bus station in an area of substantial homelessness and destitution, and (3) a prostitution hotspot in a very diverse and pedestrian part of town.

In addition to single plots, the `base_layer` argument to `ggmap` or `qmap` allows for faceted plots (see Figure 10). This is particularly useful for spatiotemporal data with discrete temporal components (day, month, season, year, etc.).

This last plot displays one of the known issues with contour plots in **ggplot2**—a "clipping" or "tearing" of the contours. Aside from that fact (which will likely be fixed in subsequent **ggplot2** versions), we can see that in fact most violent crimes happen on Monday, with a distant second being Friday. Friday's pattern is easily recognizable—a small elevated dot in the downtown bar district and an expanded region to the southwest in the district known as midtown, which has an active nightlife. Monday's pattern is not as easily explainable.

```
houston <- get_map(location = "houston", zoom = 14, color = "bw",
                   source = "osm")

HoustonMap <- ggmap(houston, base_layer = ggplot(aes(x = lon, y = lat),
                    data = violent_crimes))

HoustonMap +
  stat_density2d(aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
    bins = 5, geom = "polygon",
    data = violent_crimes) +
  scale_fill_gradient(low = "black", high = "red") +
  facet_wrap(~ day)
```

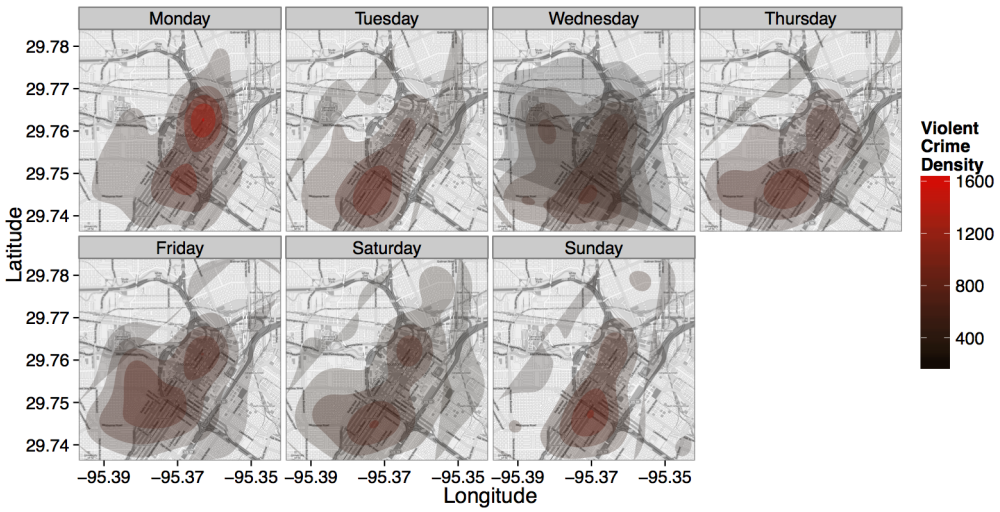


Figure 10: Faceted filled contour plot by day.

ggmap's utility functions

ggmap has several utility functions which aid in spatial exploratory data analysis.

The geocode function

The ability to move from an address to a longitude/latitude coordinate is virtually a must for visualizing spatial data. Unfortunately however, the process is almost always done outside R by using a proper geographic information system (GIS), saving the results, and importing them into R. The geocode function simplifies this process to a single line in R.

geocode is a vectorized function which accepts character strings and returns a data frame of geographic information. In the default case of `output = "simple"`, only longitudes and latitudes are returned. These are actually Mercator projections of the ubiquitous unprojected 1984 world geodetic system (WGS84), a spheroidal earth model used by Google Maps. When output is set to "more", a larger data frame is returned which provides much more Google Geocoding information on the query:

```
> geocode("baylor university", output = "more")
        lon      lat       type     loctype        address    north    south     east
1 -97.11441 31.54872 university approximate [long address] 31.55823 31.53921 -97.0984
       west postal_code       country   administrative_area_level_2
1 -97.13042       76706 united states                      mclennan
  administrative_area_level_1 locality   street  streetNo point_of_interest
1                       texas     waco s 5th st      1311              <NA>
```

In particular, administrative bodies at various levels are reported. Going further, setting `output = "all"` returns the entire JavaScript Object Notation (JSON) object given by the Google Geocoding API parsed by **rjson** (Couture-Beil, 2013).

The Geocoding API has a number of request limitations in place to prevent abuse. An unspecified short-term rate limit is in place (see mapdist below) as well as a 24-hour limit of 2,500 requests. These are monitored to some extent by the hidden global variable `.GoogleGeocodeQueryCount` and exported function geocodeQueryCheck. geocode uses these to monitor its own progress and will either (1) slow its rate depending on usage or (2) throw an error if the query limit is exceeded. Note that revgeocode shares the same request pool and is monitored by the same variable and function. To learn more about the Google Geocoding, Distance Matrix, and Directions API usage regulations, see the websites listed in the bibliography.

The revgeocode function

In some instances it is useful to convert longitude/latitude coordinates into a physical address. This is made possible (to the extent to which it is possible) with the revgeocode function which also relies on the Google Geocoding API.

```
> gc <- geocode("baylor university")
> (gc <- as.numeric(gc))
[1] -97.11441  31.54872
> revgeocode(gc)
[1] "S 1st St, Baylor University, Waco, TX 76706, USA"
```

Like geocode, more output can be provided as well –

```
> revgeocode(gc, output = "more")
        address     route     establishment  neighborhood locality
1 [long address] S 1st St Baylor University        Baylor     Waco
  administrative_area_level_2 administrative_area_level_1       country postal_code
1                   McLennan                       Texas United States       76706
```

Thus, in addition to the physical *where* of a reverse geocode (i.e., the address), revgeocode can report the *what* at various levels of granularity. Finally, an `output = "all"` option is available which returns the entire JSON object reported by Google.

The mapdist function

The ability to compute colloquial distances in a spatial setting is another invaluable commodity which typically sends analysts to a GIS. Using the Google Distance Matrix API, **ggmap** is able to provide distances for Google-determined routes for driving, bicycling, or walking. In addition to the distances, Google reports estimated travel durations as well. The full output is placed in an easy-to-use data frame. For example,

```
> from <- c("houston", "houston", "dallas")
> to <- c("waco, texas", "san antonio", "houston")
> mapdist(from, to)
     from          to      m      km    miles seconds minutes    hours
1 houston waco, texas 298004 298.004 185.1797   11907  198.45 3.307500
2 houston san antonio 320764 320.764 199.3227   11997  199.95 3.332500
3  dallas     houston 387389 387.389 240.7235   14592  243.20 4.053333
```

The default mode of transportation is driving; however, the other modes are also available. The input forms of `from` and `to` can be either physical addresses (ideal), lazy (`"the white house"`), or geographic coordinates (which are reverse geocoded). While the output defaults to the data frame format seen above, setting `output = "all"` provides the full JSON object from Google.

The Distance Matrix API limits users to 100 requests per query, 100 requests per 10 seconds, and 2500 requests per 24 hours. To the extent to which these can be easily monitored, the exported function `distQueryCheck` helps the user keep track of their remaining balance of queries. It relies on the hidden global variable `.GoogleDistQueryCount` –

```
> distQueryCheck()
2495 distance queries remaining.
> .GoogleDistQueryCount
                 time        url elements
1 2012-03-16 00:12:11 [url used]        1
2 2012-03-16 00:16:10 [url used]        2
```

If the user exceeds the limitations, `mapdist` either (1) pauses until the short-term request limit has lapsed or (2) errors if no queries are remaining. Thus, it is almost identical to the mechanism in place for geocoding. If the user believes this to be incorrect, an override is available with the `mapdist` specification `override_limit = TRUE`.

The data frame output of `mapdist` is very convenient for use with **ggplot2**. An example is provided by Figure 11, where travel times from one location ("My Office") to several nearby locations are (1) determined using `mapdist`, (2) binned into categories using `cut`, and then (3) plotted using a combination of `qmap`, `geom_text`, and `geom_rect` with the fill aesthetic set to the category of travel time. The full code is in the examples section of the documentation of `ggmap`.

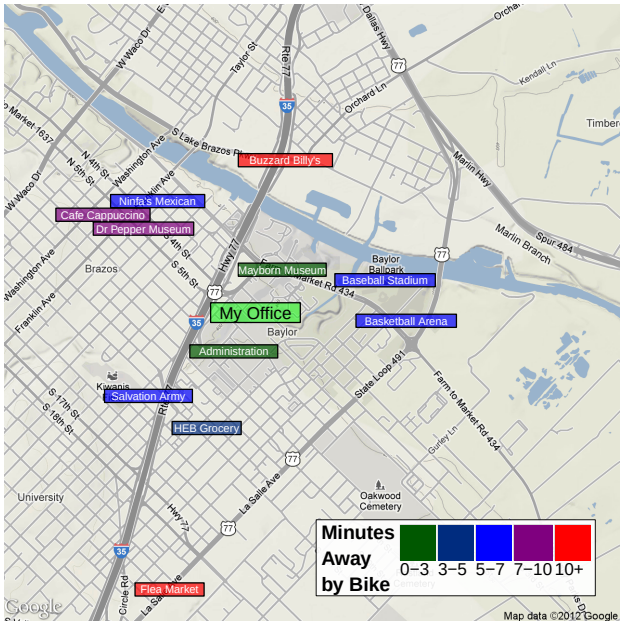


Figure 11: Distances by time provided by `mapdist`.

The `route` function

The `route` function provides the map distances for the sequence of "legs" which constitute a route between two locations. Each leg has a beginning and ending longitude/latitude coordinate along with a distance and duration in the same units as reported by `mapdist`. The collection of legs in sequence

constitutes a single route (path) most easily plotted with geom_leg, a new exported **ggplot2** geom which is simply geom_segment with rounded ends.[3]

A nice illustration of plotting routes with geom_leg can be seen in Figure 12 where three routes are plotted between the same two locations. These can be obtained using the alternatives = TRUE specification in route. alternatives requests more than one route from the origin to the destination; the returned value is again a data frame with an additional variable which serves as a route identifier (A, B, C, etc.).

```
legs_df <- route(
  'marrs mclean science, baylor university',
  '220 south 3rd street, waco, tx 76701',
  alternatives = TRUE
)

qmap('424 clay avenue, waco, tx', zoom = 15, maptype = 'hybrid',
    base_layer = ggplot(aes(x = startLon, y = startLat), data = legs_df)) +
  geom_leg(
    aes(x = startLon, y = startLat, xend = endLon, yend = endLat,
        colour = route),
    alpha = 3/4, size = 2, data = legs_df
  ) +
  labs(x = 'Longitude', y = 'Latitude', colour = 'Route') +
  facet_wrap(~ route, ncol = 3) + theme(legend.position = 'top')
```

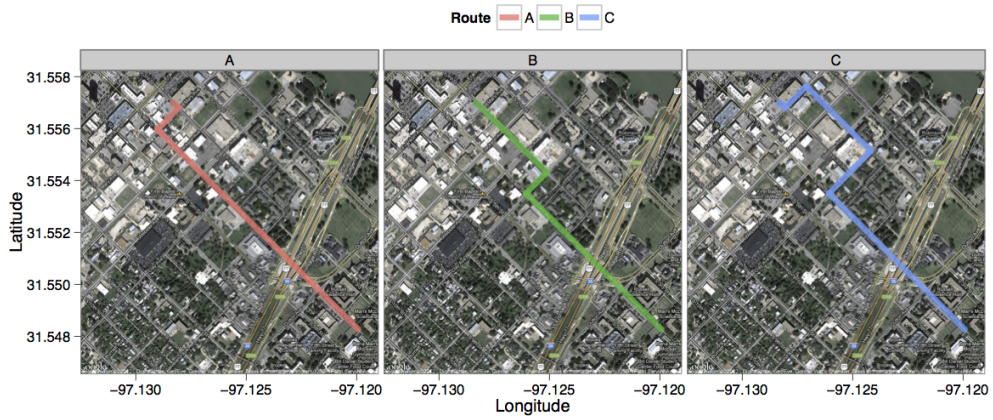


Figure 12: Three routes returned by route plotted with geom_leg.

Like map distances, there are restrictions on the limit of routes which can be requested in a given period of time as well. The number of queries left is monitored by the .GoogleRouteQueryCount variable with the routeQueryCheck function.

Plotting shape files

As a final example which is perhaps too common to ignore, plotting shape files with **ggmap** is a breeze and can be done in several ways. The easiest way is to convert the shape file to a data frame with fortify (**ggplot2**) and then add a point/path/polygon layer to the map, depending on what the contents of the shape file are (note that fortify does not yet have methods for all shape files). Additional layers can be added by simply layering on more geom layers. Figure 13 shows a basic example of plotting the U.S. Census 2000 census tracts along with complete code (U.S. Census Bureau, Geography Division, Cartographic Products Management Branch, 2001).

[3]This was inspired by the beautiful work of J. Cheshire available at http://spatialanalysis.co.uk/2012/02/great-maps-ggplot2/.

```
# get an example shape file
download.file('http://www.census.gov/geo/cob/bdy/tr/tr00shp/tr48_d00_shp.zip',
  destfile = 'census.zip')

# unzip, and load tools
unzip('census.zip'); library(maptools); library(gpclib); library(sp);
gpclibPermit()

# read data into R
shapefile <- readShapeSpatial('tr48_d00.shp',
  proj4string = CRS("+proj=longlat +datum=WGS84"))

# convert to a data.frame for use with ggplot2/ggmap and plot
data <- fortify(shapefile)
qmap('texas', zoom = 6, maptype = 'satellite') +
  geom_polygon(aes(x = long, y = lat, group = group), data = data,
    colour = 'white', fill = 'black', alpha = .4, size = .3)
```

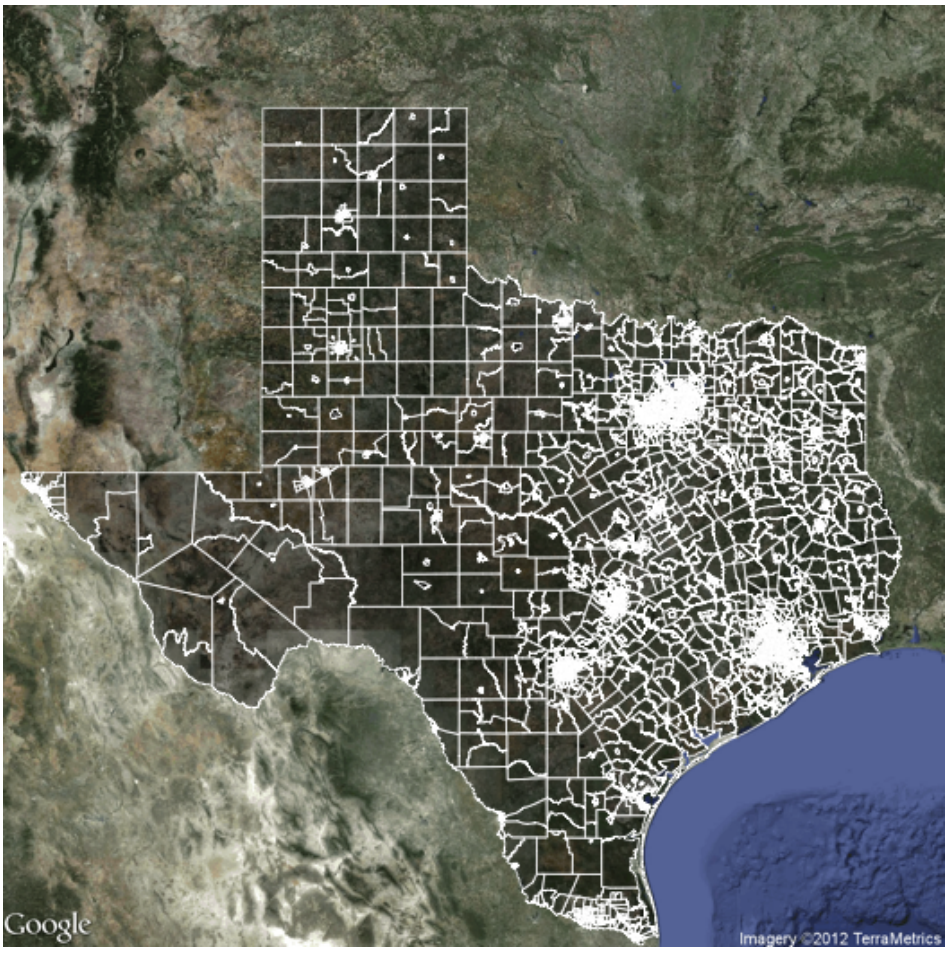


Figure 13: Plotting shape files – Census tracts in Texas from the 2000 U.S. Census.

Conclusion and future directions

Building on top of **ggplot2**, **ggmap** provides several new useful tools for visualizing spatial data. Theoretically speaking, the layered grammar of graphics attempts to enforce plotting consistency and therefore good plotting practice. Practically speaking, building **ggmap** on **ggplot2** makes the result even better as the full range of **ggplot2** capabilities can be brought to bear.

There are a number of future directions in store for **ggmap**. The new **osmar** package integrates R and the OpenStreetMap data structures with which OpenStreetMap maps, Stamen Maps, and CloudMade Maps are rendered, thereby opening a floodgate of possibilities for plotting geographic objects on top of maps or satellite imagery all within R using **ggmap** (Eugster and Schlesinger, 2013). Alternatively, integration with other spatial packages in R could provide several incredibly useful practical tools for spatial data analysis. Finally, the Google Elevation API and Places API provide additional interesting frontiers which can be incorporated into the **ggmap** framework just like the other Google APIs to give users additional capabilities though freely available geographical data.

Acknowledgment

The authors would like to thank Jean-Olivier Irisson for his comments.

Bibliography

R. A. Becker, A. R. Wilks, R. Brownrigg, and T. P. Minka. *maps: Draw Geographical Maps*, 2013. URL http://CRAN.R-project.org/package=maps. R package version 2.3-2. [p145]

R. Bivand and N. Lewin-Koh. *maptools: Tools for Reading and Handling Spatial Objects*, 2013. URL http://CRAN.R-project.org/package=maptools. R package version 0.8-23. [p145]

R. S. Bivand, E. J. Pebesma, and V. G. Rubio. *Applied Spatial Data: Analysis with R*. Springer, New York, 2008. URL http://www.asdar-book.org/. [p144]

A. Couture-Beil. *rjson: JSON for R*, 2013. URL http://CRAN.R-project.org/package=rjson. R package version 0.2.12. [p156]

M. J. A. Eugster and T. Schlesinger. osmar: Openstreetmap and R. *The R Journal*, 5(1), 2013. [p160]

I. Fellows, A. Rickett, and N. Fultz. *DeducerSpatial: A Deducer Plug-in for Spatial Data Analysis*, 2013. URL http://CRAN.R-project.org/package=DeducerSpatial. R package version 0.6. [p145]

M. Loecher and Berlin School of Economics and Law. *RgoogleMaps: Overlays on Google Map Tiles in R*, 2013. URL http://CRAN.R-project.org/package=RgoogleMaps. R package version 1.2.0.3. [p144]

E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2), 2006. URL http://cran.r-project.org/doc/Rnews/. [p144]

U.S. Census Bureau, Geography Division, Cartographic Products Management Branch. *Census 2000: Census Tract Cartographic Boundary Files*, 2001. URL http://www.census.gov/geo/www/cob/tr2000.html. [p158]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, New York, 2009. URL http://had.co.nz/ggplot2/book. [p144]

H. Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1): 3–28, 2010. [p144, 145]

H. Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1): 1–29, 2011. URL http://www.jstatsoft.org/v40/i01/. [p151]

L. Wilkinson. *The Grammar of Graphics*. Springer, New York, 2005. [p145]

David Kahle
Baylor University
Department of Statistical Science
One Bear Place #97140

Waco, TX 77005
david_kahle@baylor.edu

Hadley Wickham
Rice University
Department of Statistics, MS 138
6100 Main St.
Houston, TX 77005
hadley@rice.edu

mpoly: Multivariate Polynomials in R

by David Kahle

Abstract The **mpoly** package is a general purpose collection of tools for symbolic computing with multivariate polynomials in R. In addition to basic arithmetic, **mpoly** can take derivatives of polynomials, compute Gröbner bases of collections of polynomials, and convert polynomials into a functional form to be evaluated. Among other things, it is hoped that **mpoly** will provide an R-based foundation for the computational needs of algebraic statisticians.

Introduction

At a basic level, R is not designed for symbolic computing. Unlike computer algebra systems founded on rational arithmetic and representations like Mathematica (Wolfram Research, 2012) and Maple (Maplesoft, 2012), R's dependence on floating-point arithmetic naturally lends itself to its purpose—data analysis, numerical optimization, large(ish) data and so on. Nevertheless, the advent of algebraic statistics and its contributions to asymptotic theory in statistical models, experimental design, multiway contingency tables, and disclosure limitation has increased the need for R – as a practical tool for applied statistics – to be able to do some relatively basic operations and routines with multivariate polynomials (Diaconis and Sturmfels, 1998; Drton et al., 2009). **mpoly** is designed to try to fill this void entirely within R (Kahle, 2012). This article is intended to demonstrate its capabilities to a discipline/specialty-neutral audience.

A bit more specifically, the **mpoly** package is a general purpose collection of tools for symbolic computing with multivariate polynomials in R. While new, it is not the first package proposed to deal with multivariate polynomials. We therefore begin with a discussion of the current package intended to fulfill this need, **multipol**, in order to motivate the need for and capabilities of **mpoly** (Hankin, 2010).

Background: the multipol package

R currently has three packages which deal with polynomials – **polynom**, **PolynomF**, and **multipol**. The first two (the second being a reboot of the first) deal exclusively with univariate polynomials and as such are not suitable for the general purpose needs cited in the introduction (Venables et al., 2009; Venables, 2010). The third, **multipol**, implements multivariate polynomials in a natural way suitable for some traditional applications (e.g. simple interactions in generalized linear models) but exhibits two limitations which impede its more general use and suggest a fresh re-envisioning of multivariate polynomials in R is needed (Hankin, 2008). The first limitation has to do with the impossibility of polynomial arithmetic, and the second has to do with storing sparse polynomials.

multipol is an implementation of multivariate polynomials based on the S3 class object `"multipol"`, an unnamed multidimensional array. Consequently, **multipol** is fundamentally dependent on arrays as its basic data structure. This is evident in the construction of a `"multipol"` object:

```
> library("multipol")
Loading required package: abind

Attaching package: 'multipol'

The following object(s) are masked from
  'package:base':

    single

> a <- as.multipol(array(1:12, c(2,3,2)))
> a
, , z^0

    y^0 y^1 y^2
x^0   1   3   5
x^1   2   4   6

, , z^1
```

```
    y^0 y^1 y^2
x^0   7   9  11
x^1   8  10  12
```

In normal notation, a moment's consideration reveals that the polynomial defined above is

$$1 + 2x + 3y + 4xy + 5y^2 + 6xy^2 + 7z + 8xz + 9yz + 10xyz + 11y^2z + 12xy^2z$$

in the ring $R[x, y, z]$. (A similar pretty printing can be enabled by setting `options(showchars = TRUE)`.) Thus, the elements of the array are the coefficients on the terms appearing in the polynomial. From here, the limitations are immediately accessible.

1. Notice that the definition of a multivariate polynomial in the example above makes no reference to the variable names `x` and `y`—they are determined when the `print` method is called. In other words, a `"multipol"` object has no notion of variables whatsoever, the variables are only generated when asked for via the `print` method for class `"multipol"`. This has profound implications on polynomial arithmetic in **multipol**. In particular, arithmetic is only *possible* if two polynomials have the same number of dimensions (variables), and is only *meaningful* if those dimensions represent the same variables in the same order, the latter of which is never checked because there are no names specified which can be checked.

For example, suppose one wants to add or multiply the polynomials

$$a = 1 + 2x + 3y + 4xy \tag{1}$$

and

$$b = 1 + 2x + 3y + 4xy + 5z + 6xz + 7yz + 8xyz, \tag{2}$$

both in $R[x, y, z]$. Mathematically speaking, it is obvious that the sums and products are well-defined. However, they cannot be computed because they have a different number of variables:

```
> a <- multipol( array(1:4, c(2,2)) )
> b <- multipol( array(1:8, c(2,2,2)) )
> a + b
Error: length(dima) == length(dimb) is not TRUE
> a * b
Error: length(dim(a)) == length(dim(b)) is not TRUE
```

If the polynomials have the same number of variables, the arithmetic may be incorrect if those variables are different:

```
> options(showchars = TRUE)
> ( a <- multipol( as.array( c("x^0" = 0, "x^1" = 1) )))
[1] 1*x^1
> ( b <- multipol( as.array( c("y^0" = 0, "y^1" = 1) )))
[1] 1*x^1
> a + b
[1] 2*x^1
```

The result would seem to indicate that $a + b = x + y = 2x$, which is clearly in error. The location of the problem can easily be found however since we printed the polynomials at each step—regardless of our attempt to force (in a simple way) the labeling of the variables, we achieve the same incorrect result.

2. The array representation does not allow for sparse polynomials. For our purposes, a *sparse (multivariate) polynomial* of multidegree $\boldsymbol{d}$ is one which has "few" terms $c_{\boldsymbol{d}'}\boldsymbol{x}^{\boldsymbol{d}'}$ with multidegree $\boldsymbol{d}' \leq \boldsymbol{d}$ (element-wise) with nonzero coefficients. As an example, consider the polynomial

$$ab^2 + bc^2 + cd^2 + \cdots + yz^2 + za^2. \tag{3}$$

Since **multipol** represents multivariate polynomials with arrays, the representation requires a 26 dimensional array (a dimension for each variable) each with three levels (e.g. `a^0`, `a^1`, `a^2`). This amounts to an array with $3^{26} = 2,541,865,828,329$ cells, all but 26 of which are nonzero—a whopping 20.33TB of space if the coefficients are stored in a double-precision floating point format.

This section has introduced two shortcomings of **multipol** which significantly limit its potential use for most practical applications. It should be noted, however, that **multipol** was not intended for use with "large" polynomials. It was built with enumerative combinatorics in mind, and its inefficiency with larger polynomials was readily acknowledged in Hankin (2008), where it is suggested that perhaps a sparse array representation (as is available in Mathematica) or outsourcing to C++ routines

could alleviate the burdens of the array representation. Its inefficiency for different purposes should therefore not be considered a flaw so much as outside the package's scope. Nevertheless, the arithmetic and storage problems caused by the array representation are fatal limitations for any problem which deals with large polynomial rings (i.e., ones with many variables). Enter **mpoly**.

mpoly – the basics

mpoly is a complete reenvisioning of how multivariate polynomials and symbolic computing with multivariate polynomials should be implemented in R. Unlike **multipol**, **mpoly** uses as its most basic data structure the list. This fundamental change allows us to dispense of both issues with **multipol** discussed in the previous subsection.

In **mpoly**, an `"mpoly"` object is an S3 class object which is a list. The elements of the list are each named numeric vectors, with unique names including `coef`. Naturally, each element of an `"mpoly"` object corresponds to a term in the polynomial. The names of the elements correspond to the variables, and the values to the degrees; `coef` is the coefficient of the term. Constructing an `"mpoly"` object in this way is very straightforward using the constructor `mpoly`.

```
> library("mpoly")
> termList <- list(
+   c(coef = 1),
+   c(x = 10, coef = 2),
+   c(x = 2, coef = 3),
+   c(y = 5, coef = 4),
+   c(x = 1, y = 1, coef = 5))
> termList
[[1]]
coef
   1
[[2]]
   x coef
  10    2
[[3]]
   x coef
   2    3
[[4]]
   y coef
   5    4
[[5]]
   x    y coef
   1    1    5
> poly <- mpoly(termList)
> class(poly)
[1] "mpoly"
```

Unlike multivariate polynomials in **multipol**, those in **mpoly** not only have variable names but also an intrinsic variable order which is taken to be the unique names of elements of the `"mpoly"` object minus `coef`.[1] This can be seen with the `vars` function.

```
> vars(poly)
[1] "x" "y"
```

Viewing a multivariate polynomial as a list is a cumbersome task. To make things easier, a `print` method for `"mpoly"` objects exists and is dispatched when the object is queried by itself.

```
> poly
1  +  2 x^10  +  3 x^2  +  4 y^5  +  5 x y
```

One of the important considerations in polynomial algebra is the ordering of the terms of a multivariate polynomial. Notice the order of the terms presented in the printed version of the `"mpoly"` object; it is the order in which the terms are coded in the `"mpoly"` object itself. This can be changed in either of two ways.

First, it can be changed via the `print` method, which accepts arguments `order` for the total order used (lexicographic, graded lexicographic, and graded reverse lexicographic) and `varorder` for a

[1] To be clear, this is in the `unique(names(unlist(mpoly)))` sense; the order of the terms matters when determining the intrinsic order.

variable order different than the intrinsic order. When an order is requested but a variable order is not specified, the method messages the user to alert them to the intrinsic variable order being used.[2]

```
> print(poly, order = "lex")
using variable ordering - x, y
2 x^10  +  3 x^2  +  5 x y  +  4 y^5  +  1
> print(poly, order = "grlex")
using variable ordering - x, y
2 x^10  +  4 y^5  +  3 x^2  +  5 x y  +  1
> print(poly, order = "lex", varorder = c("y", "x"))
4 y^5  +  5 y x  +  2 x^10  +  3 x^2  +  1
> print(poly, order = "glex", varorder = c("y", "x"))
2 x^10  +  4 y^5  +  5 y x  +  3 x^2  +  1
```

Second, the elements of the "mpoly" object can be reordered to create a new "mpoly" object using the reorder method.

```
> poly
1  +  2 x^10  +  3 x^2  +  4 y^5  +  5 x y
> (poly2 <- reorder(poly, order = "lex"))
using variable ordering - x, y
2 x^10  +  3 x^2  +  5 x y  +  4 y^5  +  1
> unclass(poly2)
[[1]]
   x coef
  10    2
[[2]]
   x coef
   2    3
[[3]]
   x    y coef
   1    1    5
[[4]]
   y coef
   5    4
[[5]]
coef
   1
```

Defining polynomials: mpoly vs mp

The major workhorse of the package is the constructor mpoly itself. In particular, polynomial *reduction* (combining of like terms) and *regularization* (combining of coefficients and like variables within terms) are both performed when the multivariate polynomials are constructed with mpoly.

```
> termList <- list( c(x = 1, coef = 1), c(x = 1, coef = 2) )
> mpoly(termList) # x + 2x
3 x
> termList <- list( c(x = 5, x = 2, coef = 5, coef = 6, y = 0) )
> mpoly(termList) # x^5 * x^2 * 5 * 6 * y^0
30 x^7
```

While the constructor mpoly is nice, it is inconvenient to have to keep specifying lists in order to define polynomials. The mp function was designed for this purpose and is intended to make defining multivariate polynomials quick and easy by taking them in as character strings and parsing them into "mpoly" objects.

```
> ( p <- mp("10 x + 2 y 3 + x^2 5 y") )
10 x  +  6 y  +  5 x^2 y
> is(p, "mpoly")
[1] TRUE
> unclass(p)
```

[2]This is a subtle point. It is very possible that a polynomial in the ring $R[x, y]$ is coded with the intrinsic order $y > x$ and that, as a consequence, the typical lexicographic order will not be the one intended. The messaging is used to make clear what order is being used.

```
[[1]]
   x coef
   1   10
[[2]]
   y coef
   1    6
[[3]]
   x    y coef
   2    1    5
```

This parsing is a nontrivial process and depends heavily on the specification of the polynomial in the string. The `mp` function must first determine the variables that the user is specifying (which must be separated by spaces for disambiguation) and then construct the list to send to `mpoly` to construct the object. Because it is passed through `mpoly`, the `"mpoly"` object returned by `mp` is reduced and regularized.

```
> mp("x^2 + 10 x 6 x + 10 x 6 x y y 2")
61 x^2  +  120 x^2 y^2
```

Arithmetic, calculus, and algebra

mpoly has much more to offer than simply defining polynomials. Methods are available for addition, subtraction, multiplication, exponentiation and equality as well. Moreover, since `"mpoly"` objects know their variable names intrinsically, we can perform arithmetic with whichever polynomials we like. For example, the arithmetic with a and b from (1) and (2) is easy –

```
> a <- mp("1 + 2 x + 3 y + 4 x y")
> b <- mp("1 + 2 x + 3 y + 4 x y + 5 z + 6 x z + 7 y z + 8 x y z")
> a + b
2  +  4 x  +  6 y  +  8 x y  +  5 z  +  6 x z  +  7 y z  +  8 x y z
> b - a
5 z  +  6 x z  +  7 y z  +  8 x y z
> a * b
1  +  4 x  +  6 y  +  20 x y  +  5 z  + 16 x z  +  22 y z  +  60 x y z  +  4 x^2
 +  16 x^2 y  +  12 x^2 z  +  40 x^2 y z  +  9 y^2  +  24 x y^2  +  21 y^2 z
 +  52 x y^2 z  +  16 x^2 y^2  +  32 x^2 y^2 z
```

Exponentiation and equality are also available.

```
> a^2
1  +  4 x  +  6 y  +  20 x y  +  4 x^2  +  16 x^2 y  +  9 y^2  +  24 x y^2
 +  16 x^2 y^2
> a == b
[1] FALSE
> ( c <- mpoly(a[c(2,1,4,3)]) ) # reorder a
4 y x  +  3 y  +  2 x  +  1
> a == c
[1] TRUE
```

Here also each of the results are reduced and regularized. While the computations are done entirely in R, they are quite efficient; each of the above calculations is virtually instantaneous.

But **mpoly** does not stop there. The basic operations of the differential calculus, partial differentiation and gradients, are also available to the user and are efficient. A `deriv` method exists for `"mpoly"` objects which can be dispatched,[3] and the `gradient` function is built on `deriv` to compute gradients.

```
> deriv(b, "x")
8 y z  +  4 y  +  6 z  +  2
> gradient(b)
8 y z  +  4 y  +  6 z  +  2
8 x z  +  4 x  +  7 z  +  3
8 x y  +  6 x  +  7 y  +  5
```

The gradient is a good example of another class object in the **mpoly** package, the `"mpolyList"`. `"mpolyList"` objects are simply lists of `"mpoly"` objects and are used to hold vectors of multivariate polynomials. They can be easily specified using the `mp` function on a vector of character strings.

[3] `deriv` does not call the default method from **stats**.

```
> ( ps <- mp(c("x + y + z", "x + z^2")) )
x  +  y  +  z
x  +  z^2
> str(ps, 1)
List of 2
 $ :List of 3
  ..- attr(*, "class")= chr "mpoly"
 $ :List of 2
  ..- attr(*, "class")= chr "mpoly"
 - attr(*, "class")= chr "mpolyList"
```

The viewing of `"mpolyList"` objects is made possible by a `print` method for `"mpolyList"` objects just like for `"mpoly"` objects. Moreover addition, subtraction, and multiplication are defined for `"mpolyList"` objects as well; they each operate element-wise.

In algebraic geometry, a Gröbner basis of a polynomial ideal (or collection of polynomials generating an ideal) is a collection of polynomials which generates the same ideal and exhibits other nice properties, such as zero remainders for polynomials in the ideal and unique remainders for those outside it. They are foundational to the study of polynomial ideals and their associated varieties. In addition to differentiation, **mpoly** can also compute Gröbner bases of collections of multivariate polynomials (`"mpolyList"`) by passing the proper objects in the proper syntax to the **rSymPy** package which has an implementation of Buchberger's algorithm (Grothendieck and Bellosta, 2010).[4] The computations are performed by a Java based Python implementation and are quite fast once the Java Virtual Machine (JVM) has been initialized. The Gröbner basis is then returned as an `"mpolyList"` object.

```
> polys <- mp(c("t^4 - x", "t^3 - y", "t^2 - z"))
> gb <- grobner(polys)
using variable ordering - t, x, y, z
Loading required package: rJava
> gb
-1 z  +  t^2
t y  -  z^2
-1 y  +  z t
x  -  z^2
y^2  -  z^3
> class(gb)
[1] "mpolyList"
```

Moreover, `grobner` can calculate Gröbner bases with respect to various monomial orders and any variable ordering.

```
> polys <- mp(c("x^2 - 2 y^2", "x y - 3"))
> grobner(polys, varorder = c("x", "y"))
3 x  -  2 y^3
-9  +  2 y^4
> grobner(polys, varorder = c("x", "y"), order = "grlex")
-3 x  +  2 y^3
x^2  -  2 y^2
-3  +  x y
```

The task of computing Gröbner bases is the only job **mpoly** outsources from R. This is because implementations of Gröbner bases algorithms are (1) complex and highly prone to error, (2) scarce and therefore difficult to emulate, and (3) highly iterative; thus it was thought best to leave the routine to more expert programmers and frameworks outside R. Unfortunately, there is currently no Gröbner walk algorithm available to convert a Gröbner basis in one monomial order to a Gröbner basis in another, a technique often used to quickly compute Gröbner bases in more difficult orders (e.g. lexicographic) from "easier" ones (e.g. graded reverse lexicographic), so there are still a number of improvements which can be made.

Evaluating polynomials

Apart from being interesting algebraic objects, polynomials can of course be thought of as functions. To access this functional perspective, we can convert an `"mpoly"` or `"mpolyList"` object to a `function`

[4] Buchberger's algorithm is the standard method of calculating Gröbner bases, however, faster methods are known.

using the "mpoly" method of as.function. This is particularly suited to R's strengths since R is geared towards evaluation and numerical optimization

```
> library("ggplot2"); theme_set(theme_bw())
> ( p <- mp("x") * mp("x - .5") * mp("x -  1") ) # x(x-.5)(x-1)
x^3  -  1.5 x^2  +  0.5 x
> f <- as.function(p)
f(x)
> f
function(x){x**3  -  1.5 * x**2  +  0.5 * x}
<environment: 0x1218bc270>
> f(10)
[1] 855
> s <- seq(-.1, 1.1, length.out = 201)
> df <- data.frame(x = s, y = f(s))
> qplot(x, y, data = df, geom = "path") + geom_hline(yintercept = 0, colour = I("red"))
```

The plot generated is included in Figure 1, where one can see that the function has the correct zeros. For multivariate polynomials the syntax is the same, and as.function can provide the function with

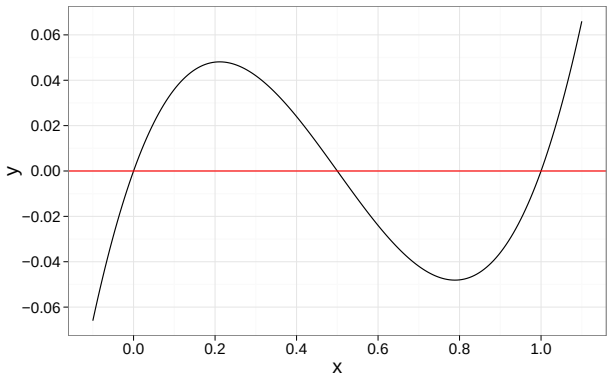


Figure 1: The as.function method for "mpoly" objects.

(1) a vector argument (ideal for passing to optimization routines such as optim) or (2) a sequence of arguments.

```
> mpoly <- mp("x + 3 x y + z^2 x")
>
> f <- as.function(mpoly)
f(.) with . = (x, y, z)
> f(1:3)
[1] 16
>
> f <- as.function(mpoly, vector = FALSE)
f(x, y, z)
> f(1, 2, 3)
[1] 16
```

"mpolyList" objects can be converted into functions in the same way. Note that in both cases the user is messaged with the appropriate syntax for the resulting function.

```
> polys <- mp(c("x + 1", "y^2 + z"))
> f <- as.function(polys)
f(.) with . = (x, y, z)
> f(1:3)
[1] 2 7
```

Another neat example for illustrating this is visualizing Rosenbrock's "banana" function,

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2,$$

which is a common test example for optimization routines (Figure 2).

```
> f <- mp("1 - 2 x + x^2 + 100 x^4 - 200 x^2 y + 100 y^2")
> f <- as.function(f)
```

```
> df <- expand.grid(x = seq(-2, 2, .01), y = seq(-1, 3, .01))
> df$f <- apply(df, 1, f)
> library("scales")
> qplot(x, y, data = df, geom = c("raster", "contour"),
+    fill = f + .001, z = f, colour = I("white"), bins = 6) +
+  scale_fill_gradient2(
+    low = "blue", mid = "green", high = "red",
+    trans = "log10", guide = "none"
+  )
```

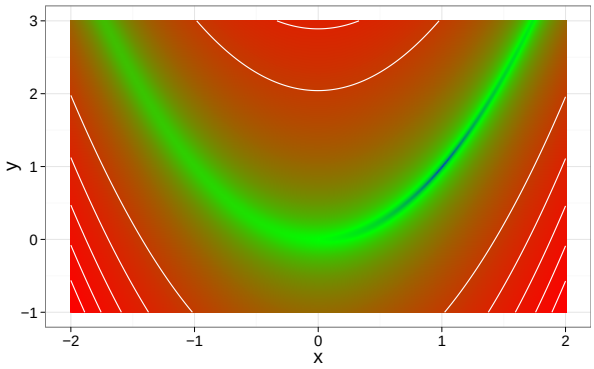


Figure 2: The Rosenbrock function plotted by evaluating an "mpoly" object via as.function and apply with vector arguments.

Conclusion and future directions

While other functionality is also provided (such as a terms method, a function to compute least common multiples, and some combinatorial functions), the presented ones are the main contributions of the **mpoly** package. Put together, they provide a package which is not only user-friendly, efficient, and useful for polynomial algebra, but also a solid foundation upon which further developments can be made to make polynomials more accessible in R. In particular, it provides a nice starting point for any future package dealing with algebraic statistics.

There are, however, several improvements which can be made. First, the primary constructor function mpoly can be made significantly faster by outsourcing its job to C++, as previously suggested (Hankin, 2008). Second, improvements can be made to speed up some of the arithmetic, for example addition-chain exponentiation for polynomial exponents (as opposed to iterative multiplication). Last, improvements can be made into the flexibility of the facilitated constructor mp. In particular, parenthetical expressions are not currently handled but would be of great practical use.

Acknowledgments

The author would like to thank Martyn Plummer and two anonymous reviewers for useful suggestions which made this and future work better.

Bibliography

P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. *The Annals of Statistics*, 26(1):363–397, 1998. [p162]

M. Drton, B. Sturmfels, and S. Sullivant. *Lectures on Algebraic Statistics*. Birkhäuser, 2009. [p162]

G. Grothendieck and C. J. G. Bellosta. *rSymPy: R Interface to SymPy Computer Algebra System*, 2010. URL http://CRAN.R-project.org/package=rSymPy. R package version 0.2-1.1. [p167]

R. K. S. Hankin. Programmers' niche: Multivariate polynomials in R. *R News*, 8(1):41–45, 2008. [p162, 163, 169]

R. K. S. Hankin. *multipol: Multivariate Polynomials*, 2010. URL http://CRAN.R-project.org/package=multipol. R package version 1.0-4. [p162]

D. Kahle. *mpoly: Symbolic Computation and More with Multivariate Polynomials*, 2012. URL http://CRAN.R-project.org/package=mpoly. R package version 0.0.1. [p162]

Maplesoft. *Maple 16*. Waterloo, Canada, 2012. URL http://www.maplesoft.com/. [p162]

B. Venables. *PolynomF: Polynomials in R*, 2010. URL http://CRAN.R-project.org/package=PolynomF. R package version 0.94. [p162]

B. Venables, K. Hornik, and M. Maechler. *polynom: A Collection of Functions to Implement a Class for Univariate Polynomial Manipulations*, 2009. URL http://CRAN.R-project.org/package=polynom. R package version 1.3-6. [p162]

Wolfram Research. *Mathematica 9.0*. Champaign, 2012. URL http://www.wolfram.com/. [p162]

David Kahle
Baylor University
Department of Statistical Science
One Bear Place #97140
Waco, TX 76798
david_kahle@baylor.edu

beadarrayFilter: An R Package to Filter Beads

by Anyiawung Chiara Forcheh, Geert Verbeke, Adetayo Kasim, Dan Lin, Ziv Shkedy, Willem Talloen, Hinrich W.H. Göhlmann and Lieven Clement

Abstract Microarrays enable the expression levels of thousands of genes to be measured simultaneously. However, only a small fraction of these genes are expected to be expressed under different experimental conditions. Nowadays, filtering has been introduced as a step in the microarray preprocessing pipeline. Gene filtering aims at reducing the dimensionality of data by filtering redundant features prior to the actual statistical analysis. Previous filtering methods focus on the Affymetrix platform and can not be easily ported to the Illumina platform. As such, we developed a filtering method for Illumina bead arrays. We developed an R package, **beadarrayFilter**, to implement the latter method. In this paper, the main functions in the package are highlighted and using many examples, we illustrate how **beadarrayFilter** can be used to filter bead arrays.

Introduction

Gene expression patterns are commonly assessed by microarrays that can measure thousands of genes simultaneously. However, in a typical microarray experiment, only a small fraction of the genes are informative which motivated the development of gene filtering methods. Gene filtering aims at reducing the dimensionality of data by filtering redundant features prior to the actual statistical analysis. This has been shown to improve differential expression analysis with Affymetrix microarrays (e.g., Talloen et al., 2007; Calza et al., 2007; Kasim et al., 2010).

Although different microarrays platforms share the same principle of hybridizing DNA to a complementary probe, they differ considerably by design. Unlike the Affymetrix microarrays which have sets of unique probes targeting a particular gene (resulting in a probe set for a targeted gene), Illumina microarrays have sets of identical probes. Thus, the existing filtering methods can not be readily ported to the Illumina platform. As a result, Forcheh et al. (2012) developed a filtering method for Illumina bead arrays. Forcheh et al. (2012) equally showed that filtering improves the analysis of differential expression. We provide the implementation of their method in the **beadarrayFilter** R software package. The **beadarrayFilter** package can take a normalized data frame or a normalized bead array ExpressionSetIllumina object (obtained using the `summarize` or `readBeadSummaryData` functions in the Bioconductor package **beadarray** by Dunning et al., 2007) or a normalized LumiBatch object as input and returns a list containing a filtered data frame or a filtered bead array ExpressionSetIllumina object or a filtered LumiBatch object, respectively. The package can also process summarized and normalized average intensities (eSet), their standard errors (seSet) and the number of beads used for summarization (nSet) as input and returns a list of components including the intra-cluster correlations (ICC), which can be used to assess different filtering strategies.

The paper contains a brief background of the filtering methodology followed by the introduction of the **beadarrayFilter** package with illustrative examples.

Filtering Illumina bead types

Let Y_{kij} be the bead-level expression intensity of bead j in sample (array) i in treatment group k, $i = 1, \ldots, n_k$, $j = 1, \ldots, m_{ki}$, $k = 1, \ldots, K$, then, Forcheh et al. (2012) proposed the following filtering model:

$$Y_{kij} = \mu + b_i + \epsilon_{kij}, \tag{1}$$

where μ represents the average expression for a bead type across all samples, b_i is the array specific random effect and ϵ_{kij} are the measurement errors, both normally distributed with mean zero and variance τ^2 and σ^2, respectively:

$$b_i \sim N(0, \tau^2), \epsilon_{kij} \sim N(0, \sigma^2).$$

The τ^2 captures the between-array variability while σ^2 models the within-array variability. Model (1) implies a common ICC, (Verbeke and Molenberghs, 2000), given by

$$\rho = \frac{\tau^2}{\tau^2 + \sigma^2}, \tag{2}$$

which is the correlation among the replicate probes on the array.

An informative bead type is expected to have a relatively high value of ρ since all corresponding beads have the same sequence. As such, the between-array variability is the dominant variance component for informative bead types while the within-array variability is the largest variance component for the non-informative bead types.

For Illumina bead arrays, within-array variability is expected to vary across all arrays and treatment groups in the experiment. In this regard, Forcheh et al. (2012) included an array/treatment specific variance component to adjust the model for heteroscedasticity i.e., $\epsilon_{kij} \sim N(0, \sigma^2_{ki})$. Hence, the ICC becomes bead/array/group specific,

$$\rho_{ki} = \frac{\tau^2}{\tau^2 + \sigma^2_{ki}}, \tag{3}$$

and $\sum_{k=1}^{K} n_k$ ICCs are obtained for each bead type (more details can be found in Forcheh et al., 2012).

ICC based filtering of Affymetrix microarray data has been proposed in the literature (e.g., Talloen et al., 2007; Kasim et al., 2010). Typically, an ICC cut-off $\delta = 0.5$ has been used to declare a transcript informative. Forcheh et al. (2012) also relied on the ICC as a filtering criterion and proposed different thresholding schemes based on the five number summaries, i.e., a bead type can be called informative based on thresholding (1) the minimum ICC, (2) 25, (3) 50, (4) 75 quantiles or (5) the maximum ICC of all the arrays. Filtering is expected to become less stringent from strategy (1)–(5). These different thresholding strategies are implemented within the package.

beadarrayFilter package

The **beadarrayFilter** package can either take a normalized ExpressionSetIllumina object, a normalized LumiBatch object, a normalized data.frame or a normalized eSet, seSet and nSet as input and returns a list. We refer the user to the documentation of the Bioconductor packages **beadarray** and **lumi** for more details on generating ExpressionSetIllumina objects or LumiBatch objects. For each bead type, the ICCs can be summarized using the 5 number summary or user specified quantiles. The corresponding ICC summaries are used for obtaining informative bead types. The package contains two major functions, which we refer to as: (1) a low level function `iccFun` and (2) a wrapper function `beadtypeFilter`. Model fitting is done using a modified version of the `MLM.beadarray` function of Kim and Lin (2011). Details on the functions can be obtained by using the help function in R (`?beadtypeFilter` or `?iccFun`).

Installation of beadarrayFilter

The **beadarrayFilter** package is available at CRAN, and can be installed using `install.packages("beadarrayFilter")`.

`beadtypeFilter` function

The `beadtypeFilter` function is a wrapper function for the `iccFun` function and is designed for users with a primary interest in obtaining filtered bead types. This function takes a normalized ExpressionSetIllumina object, a normalized LumiBatch object or a normalized data.frame and returns the names of the informative bead types. Optionally, the filtered ExpressionSetIllumina object or the filtered data.frame can also be returned. The filtered ExpressionSetIllumina object, or filtered LumiBatch object or the filtered data.frame can then be used for the downstream analysis.

`iccFun` function

`iccFun` is a low level function. It is designed for users who want to assess different filtering strategies. It takes a normalized eSet, seSet and nSet and the bead types identification variable (ProbeID), fits the filtering Model (1), calculates the ICC for each bead type on each array/treatment group, summaries the ICCs at the specified quantiles, and returns the ICC summaries, the within-array variances, the between-array variances as well as all ICCs. The ICCs output can later be used for filtering or to assess different filtering strategies. Note the information printed as you execute the `beadtypeFilter` or `iccFun` functions:

- [1] "Number of converged transcripts: ... "
  This indicates the number of transcripts for which the filtering model has already converged while "Now ... remaining..." tells the number of transcripts still to be processed.

- [1] "Computing ICC for each array"
  This message is printed when the function begins to calculate the ICC for each array once the filtering model has been fitted to all the transcripts.
- [1] "Summarising the ICC at supplied quantile(s)"
  This message indicates the last stage of the filtering function where the ICCs are summarized at the supplied quantiles (see Forcheh et al., 2012).

The `emCDF` function within the **beadarrayFilter** package is used to plot the empirical cumulative density functions (edcfs) for the different threshold strategies discussed above. It processes the `iccFun` output and plots the empirical cumulative density functions (ecdf) for the different threshold strategies as discussed in Forcheh et al. (2012). It is expected that the filtering becomes more stringent as the ICC threshold increases and/or as the the thresholded ICC quantile decreases.

Informative bead types will have larger between-array variances as compared to the within-array variances. The `varianceplot` function takes the estimated between- and within-array variances from the `iccFun` function as inputs and plots them. The variances of noninformative bead types are plotted in blue while those of the informative bead types are displayed in red.

Upon filtering the `MLM.beadarray` function can be used for downstream analysis of single factor designs. For more details, see Kim and Lin (2011).

A summary of the functions within the **beadarrayFilter** package and their use is presented in Table 1.

Table 1: Main Functions within the **beadarrayFilter** package.

Function	Description
`beadtypeFilter()`	Fits the filtering model as in Forcheh et al. (2012), computes the ICC and filters the bead types.
`iccFun()`	Fits the filtering model as in Forcheh et al. (2012), and computes the ICC which can later be used for filtering or to assess different filtering strategies
`MLM.beadarray()`	Function to fit the filtering model or for the downstream analysis of single factor experimental designs
`emCDF()`	Plots the ecdf for the different threshold strategies
`varianceplot()`	Plots the the between-array and the within-array variances

Examples

The data set `exampleSummaryData` from the Bioconductor package **beadarrayExampleData** (Dunning et al., 2007), is used to illustrate filtering of ExpressionSetIllumina objects while the publicly available spike-in data (Dunning et al., 2008) are used to process data.frames. The $\log_2$ summarized expression intensities, obtained from Brain and Universal Human Reference RNA (UHRR) samples in the data set `exampleSummaryData` will be used. There are 12 arrays, 6 for the Brain samples and 6 for the UHRR samples. The spike-in dataset consists of 48 arrays and an array contained $\sim$ 48,000 bead types (non-spikes) and 33 bead types (spikes) targeting bacterial and viral genes absent in the mouse genome. We use the same subset of the spike-in data as in Kim and Lin (2011). This dataset includes 34,654 bead types targeting annotated genes with Genebank IDs and the 33 spikes. The data can be downloaded from `https://ephpublic.aecom.yu.edu/sites/rkim/Supplementary/files/KimLin2010.html` with the folder name "A normalized dataset for example analysis". We also show how the downstream analysis can be performed upon filtering using the spike-in data. Filtering of LumiBatch objects is illustrated using the BeadStudio output used in the vignette "beadsummary" from the **beadarray** package, which can be downloaded from `http://www.switchtoi.com/datasets.ilmn`.

`beadtypeFilter` function

This subsection shows how to use the `beadtypeFilter` function to filter normalized ExpressionSetIllumina objects, LumiBatch objects and data.frames. For an ExpressionSetIllumina, this is done using the

exampleSummaryData data set from the **beadarrayExampleData** package.

```
# Normalize the log2 transformed data
> library("beadarrayFilter")
> data("exampleSummaryData", package = "beadarrayExampleData")
> exampleSummaryDataNorm <- normaliseIllumina(channel(exampleSummaryData, "G"),
+    method = "quantile", transform = "none")
# Filter the ExpressionSetIllumina
> iccResults <- beadtypeFilter(exampleSummaryDataNorm, Quantile = 1,
+    keepData = TRUE, delta = 0.5)
```

By specifying `iccQuant = 1` and `delta = 0.5`, the bead types are filtered using the maximum ICC at a cutoff of 50%. The output of the `beadtypeFilter` function can then be observed as follows:

```
> head(iccResults$InformProbeNames)
[1] "ILMN_1802380" "ILMN_1736104" "ILMN_1792389" "ILMN_1705423" "ILMN_1697642"
[6] "ILMN_1788184"
> exprs(iccResults$informData)[1:6, 1:5]
             4613710017_B 4613710052_B 4613710054_B 4616443079_B 4616443093_B
ILMN_1802380     8.216547     8.229713     8.097047     8.343822     8.249190
ILMN_1736104     5.317065     5.470957     5.054653     5.100678     5.446530
ILMN_1792389     6.725049     7.003632     6.783809     7.214921     7.257032
ILMN_1705423     5.496207     4.845898     5.394206     5.422772     5.479191
ILMN_1697642     7.977234     7.912246     7.668253     7.850134     7.758535
ILMN_1788184     5.291988     5.614500     5.565426     5.473346     5.573395
> head(fData(iccResults$informData))
           ArrayAddressID   IlluminaID  Status
ILMN_1802380          10008 ILMN_1802380 regular
ILMN_1736104          10017 ILMN_1736104 regular
ILMN_1792389          10019 ILMN_1792389 regular
ILMN_1705423          10039 ILMN_1705423 regular
ILMN_1697642          10044 ILMN_1697642 regular
ILMN_1788184          10048 ILMN_1788184 regular
> dim(exampleSummaryDataNorm)
Features  Samples Channels
   49576       12        1
> dim(iccResults$informData)
Features  Samples Channels
   23419       12        1
```

23419 out of the 49576 bead types were declared informative using the maximum ICC at a cutoff point of 50%.

For a LumiBatch object, filtering is illustrated using the non-normalized data, an output of BeadStudio used in the "beadsummary" vignette from the **beadarray** package. The data file, 'AsuragenMAQC_BeadStudioOutput.zip', can be downloaded from http://www.switchtoi.com/datasets.ilmn. Once the file has been downloaded, unzip its content to your R working directory.

```
> require(lumi)

# Set the working directory to the directory where the unzipped data file was saved.
> setwd("C:/Multi_level_Illumina_feb2011/RPackageFinal/beadstudiooutputData")
# Read in the data using lumiR to obtain a LumiBatch object
> x.lumi <- lumiR("AsuragenMAQC-probe-raw.txt")
# Normalize the data without any further transformation step
> lumi.N <- lumiN(x.lumi, "rsn")
# Filter the LumiBatch
> iccResult <- beadtypeFilter(lumi.N, Quantile = 1, keepData = TRUE, delta = 0.5)
```

By specifying `iccQuant = 1` and `delta = 0.5`, the bead types are filtered using the maximum ICC at a cutoff of 50%.

```
> dim(lumi.N)
Features  Samples
   48701        6
> dim(iccResult$informData)
Features  Samples
    1195        6
```

Only 1195 of the 48701 bead types were declared informative using the maximum ICC at a cutoff of 50%. This may be due to the way the data was summarized and normalized. How the processing of bead array data affects bead types filtering is a topic of future research.

For a data.frame, the `beadtypeFilter` function is illustrated using the Illumina spike-in data. Read the data from the file location where the data had been downloaded, unzipped and saved.

```
> filepath <- "C:/Multi_level_Illumina_feb2011/log2scale.normalized.txt"
> dt <- read.delim(filepath, header = TRUE, as.is = TRUE, row.names = NULL)[,-1]
> dt[1:6,1:5]
  ProbeID X1377192001_A.AVG_Signal X1377192001_A.Detection.Pval
1   50014                 6.150486                  0.579207900
2   50017                 6.616132                  0.074257430
3   50019                 8.164317                  0.000000000
4   50020                 7.414991                  0.001856436
5   50022                 5.804593                  0.974628700
6   50025                 6.412067                  0.173267300
  X1377192001_A.Avg_NBEADS X1377192001_A.BEAD_STDERR
1                       27                0.09889349
2                       40                0.05644992
3                       25                0.06384269
4                       27                0.07853792
5                       38                0.08098911
6                       28                0.08153830
```

Note, that the data.frame supplied to the `beadtypeFilter` function should contain the summarized intensities (`eSet`), standard errors (`seSet`) and the number of beads used for the summarization (`nSet`). When using a data frame, column names should be conform to BeadStudio output, i.e., the column names for eSet should end on "Signal", those for seSet on "STDERR" and the columns corresponding to nSet should end on "NBEADS". It is preferable to use an identification variable with a unique ID for each bead type. In the spike-in data, the spikes all have the same `TargetID`, thus the `ProbeID` is preferred. Similar to the ExpressionSetIllumina example, the `beadtypeFilter` function is used for filtering.

```
> iccResults <- beadtypeFilter(dt, Quantile = 0.5, keepData = TRUE, delta = 0.5)
```

By specifying `Quantile = 0.5`, bead types are filtered using the median ICC.

```
> head(iccResults$InformProbeNames)
[1]  50280  50440  70594 110138 110685 130402
> dim(dt)
[1] 34687   193
> dim(iccResults$informData)
[1] 238 193
```

238 of the 34687 bead types were declared informative based on thresholding the median ICC at a cutoff of 50%. A large number of bead types have been filtered out. It should be noted that this is probably due to the artificial nature of the spike-in data and we would expect lesser bead types to be filtered out in real life data.

iccFun function

The examples in this section show how the `iccFun` function can be used to process different data types, observe its output and assess the filtering strategies.

Processing data using the `iccFun` function

1. **Processing a data.frame**

```
> filepath <- "C:/Multi_level_Illumina_feb2011/log2scale.normalized.txt"
> dt <- read.delim(filepath, header = TRUE, as.is = TRUE, row.names = NULL)[,-1]
> eSet <- dt[, grep("Signal", names(dt))]
> seSet <- dt[, grep("STDERR", names(dt))]
> nSet <- dt[, grep("NBEADS", names(dt))]
> ProbeID <- dt[, 2]
> iccResults <- iccFun(eSet, seSet, nSet, ProbeID = ProbeID,
                       iccQuant = c(0, 0.25, 0.5, 0.75, 0.8, 1),
                       diffIcc = TRUE, keepData = TRUE)
```

2. **Processing a LumiBatch object**

```
> setwd("C:/Multi_level_Illumina_feb2011/RPackageFinal/beadstudiooutputData")
# Read in the data using \code{lumiR} to obtain a LumiBatch object
> x.lumi <- lumiR("AsuragenMAQC-probe-raw.txt")
> lumi.N <- lumiN(x.lumi, "rsn")
> eSet <- exprs(lumi.N)
> seSet <- se.exprs(lumi.N)
> nSet <- beadNum(lumi.N)
> group <- c(1:dim(eSet)[2])
> ProbeID = fData(lumi.N)$ProbeID
> iccResults <- iccFun(eSet, seSet, nSet, ProbeID = ProbeID,
+                      iccQuant = c(0, 0.25, 0.5, 1),
+                      diffIcc = TRUE, keepData = TRUE)
```

3. **Processing an ExpressionSetIllumina object**

```
> exampleSummaryDataNorm <-
+      normaliseIllumina(channel(exampleSummaryData, "G"),
+                        method = "quantile", transform = "none")
> aaa <-
+      na.omit(data.frame(I(rownames(exprs(exampleSummaryDataNorm))),
+                         exprs(exampleSummaryDataNorm)))
> ProbeID <- aaa[, 1]
> eSet <- na.omit(exprs(exampleSummaryDataNorm))
> stddev <- na.omit(se.exprs(exampleSummaryDataNorm))
> nSet <- na.omit(attributes(exampleSummaryDataNorm)$assayData$nObservations)
> seSet <- stddev/sqrt(nSet)
> iccResults <- iccFun(eSet, seSet, nSet, ProbeID = ProbeID,
+                      iccQuant = c(0, 0.25, 0.5, 1))
```

The output of the `iccFun` function

In this subsection, we illustrate how the output of the `iccFun` function can be observed. Note that we display the results for `exampleSummaryData`, the output for the spike-in data can be found in Forcheh et al. (2012).

```
> head(iccResults$betweenvar)
      ProbeID    fit1.tau2
1 ILMN_1802380 1.3154886475
2 ILMN_1893287 0.0202718744
3 ILMN_1736104 0.7883136626
4 ILMN_1792389 0.5374776179
5 ILMN_1854015 0.0000000000
6 ILMN_1904757 0.0004272419

> iccResults$withinvar[1:6, 1:6]
             ProbeID      sigma2.4613710017_B sigma2.4613710052_B sigma2.4613710054_B
ILMN_1802380 ILMN_1802380          0.08024396           0.1133679          0.07562057
ILMN_1893287 ILMN_1893287          0.15510050           0.1495736          0.31645854
ILMN_1736104 ILMN_1736104          0.22109680           0.2449570          0.16237022
ILMN_1792389 ILMN_1792389          0.16305881           0.2232660          0.25316536
ILMN_1854015 ILMN_1854015          0.31302729           0.1367953          0.29684239
ILMN_1904757 ILMN_1904757          0.11065525           0.2427457          0.35319329
             sigma2.4616443079_B sigma2.4616443093_B
ILMN_1802380           0.1715118           0.1282700
ILMN_1893287           0.4629203           0.1956166
ILMN_1736104           0.2781976           0.2364219
ILMN_1792389           0.1187983           0.1560972
ILMN_1854015           0.2776505           0.4338320
ILMN_1904757           0.2621982           0.4215812

> head(iccResults$iccAll)
            [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
[1,] 0.942507641 0.920658321 0.945640089 0.884659197 0.911155532 0.945550561
[2,] 0.115593318 0.119354803 0.060202088 0.041954058 0.093899753 0.076838800
```

```
[3,] 0.780964425 0.762930437 0.829206926 0.739151757 0.769284994 0.672257600
[4,] 0.767237213 0.706516107 0.679798133 0.818981163 0.774938161 0.794058983
[5,] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
[6,] 0.003846168 0.001756947 0.001208193 0.001626811 0.001012401 0.003354805
            [,7]         [,8]        [,9]       [,10]       [,11]        [,12]
[1,] 0.813348960 0.2851197747 0.924933768 0.921820518 0.953747680 0.9672588084
[2,] 0.146496212 0.0086476707 0.079882094 0.046679979 0.083112294 0.0577318494
[3,] 0.862368982 0.4039671466 0.788168517 0.745235220 0.651296881 0.5468796285
[4,] 0.795226413 0.2446849630 0.818919242 0.889560842 0.859993713 0.7440351650
[5,] 0.000000000 0.0000000000 0.000000000 0.000000000 0.000000000 0.0000000000
[6,] 0.002726754 0.0005326228 0.001815729 0.005423024 0.002744552 0.0009235276
> head(iccResults$icc)
 ProbeID           q0       q0.25        q0.5          q1
1 ILMN_1802380 0.2851197747 0.904531448 0.923377143 0.967258808
2 ILMN_1893287 0.0086476707 0.054968882 0.078360447 0.146496212
3 ILMN_1736104 0.4039671466 0.667017420 0.754082829 0.862368982
4 ILMN_1792389 0.2446849630 0.734655400 0.784498572 0.889560842
5 ILMN_1854015 0.0000000000 0.000000000 0.000000000 0.000000000
6 ILMN_1904757 0.0005326228 0.001159245 0.001786338 0.005423024
```

If desired, the `iccResults$icc` output from the `iccFun` function can be used for filtering and assessing the different filtering strategies.

```
# Obtaining the number of informative bead types at each of the specified ICC quantiles
> apply(iccResults$icc[, -1], 2, function(x, thres) sum(x >= thres), thres = 0.5)
q0 q0.25  q0.5    q1
 4699 15784 17757 23419
# Obtaining the informative bead types using the minimum ICC
> filterDataNorm <- exampleSummaryDataNorm[subset(iccResults$icc,
+                                                 iccResults$icc[, 2] >= 0.5)[, 1], ]
>  dim(filterDataNorm)
Features  Samples Channels
    4699       12        1
```

Assessing the filtering strategies

This is done using the `emCDF` function (Figure 1).

```
> emCDF(iccResults, iccQuant = c(0, 0.25, 0.5, 1))
```

Further, the within- and between-array variances at the minimum ICC can be observed using the `varianceplot` function (Figure 2).

```
> varianceplot(iccResults, q = 1, delta = 0.8)
```

By specifying `q = 1` and `delta = 0.8`, the informative beads (displayed in red) are obtained using the minimum ICC at a cutoff of 80%.

Analysis of differential expression

Once the data have been filtered, they can be used for downstream analysis. Here, we assess differential expression using a filtered data.frame. For the example 608 bead types were declared informative based on the maximum ICC. We refer to the help file of the `MLM.beadarray` function in the **beadarrayFilter** package for an example on a ExpressionSetIllumina object.

```
> iccResults <- beadtypeFilter(dt, Quantile = 1, keepData = TRUE, delta = 0.5)
> dim(iccResults$informData)
> dat <- iccResults$informData
> eSet <- dat[, grep("Signal", names(dat))]
> seSet <- dat[, grep("STDERR", names(dat))]
> nSet <- dat[, grep("NBEADS", names(dat))]
```

We define the group variable to compare concentrations 0.3 and 0.1 pM in the spike-in data. This is done by selecting the column numbers of the arrays corresponding to the concentrations of interest.

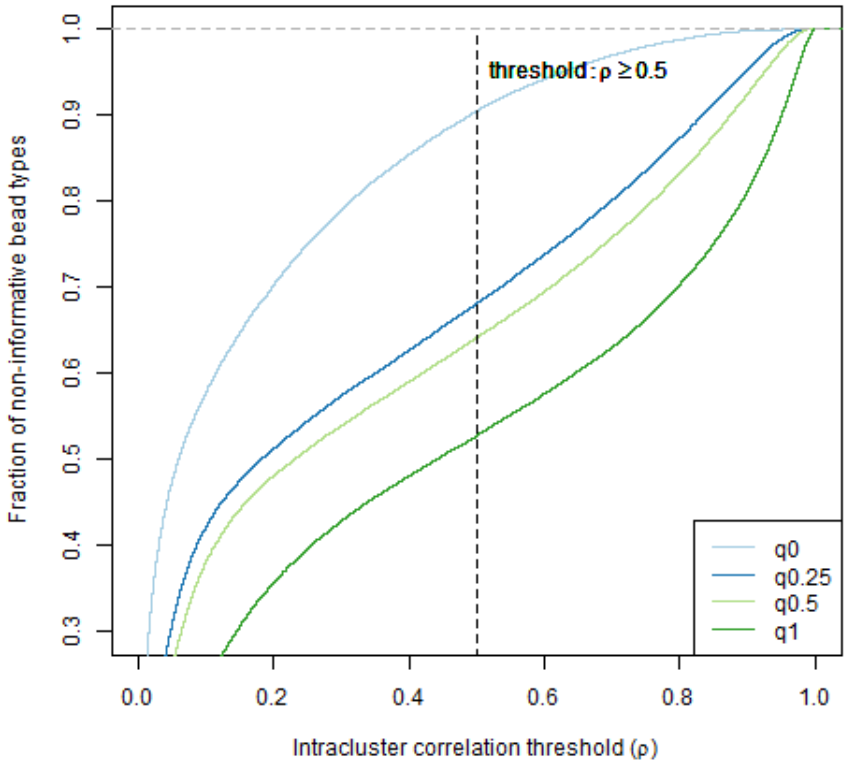


Figure 1: The empirical cumulative density function of the ICCs at the specified quantiles.

```
> group1 <- c(26, 32, 38, 44)
> group2 <- c(27, 33, 39, 45)
> fit1 <- MLM.beadarray(eSet, seSet, nSet, list(group1, group2),
+                       var.equal = TRUE, max.iteration = 20, method = "ML")
```

The output of the `MLM.beadarray` function can then be used to test for equality of mean expression between the two concentrations

```
> df <- length(group1) + length(group2) - 2
> fit1$pvalue <- 2 * (1-pt(abs(fit1$t.statistics), df))
> fit1$pvalAdjust <- p.adjust(fit1$pvalue, method = "fdr", n = length(fit1$pvalue))
> length(which(fit1$pvalAdjust < 0.05))
[1] 29
```

i.e., 29 bead types were found to be differentially expressed between concentrations 0.3 and 0.1 pM. Note that 22 of these 29 bead types are true positives (spikes).

Discussion

The **beadarrayFilter** package can be used to filter Illumina bead array data. The `beadtypeFilter` function can filter normalized ExpressionSetIllumina objects, normalized LumiBatch objects as well as normalized data.frames and returns the names of the informative bead types. Optionally, the user can also obtain the filtered data. This, however, does not return the required outputs to assess different filtering strategies nor the variances using the `emCDF` or the `varianceplot` functions, respectively. The `iccFun` function can be used to customize filtering strategies. It returns the required outputs for assessing different filtering strategies and the between- and within-array variances.

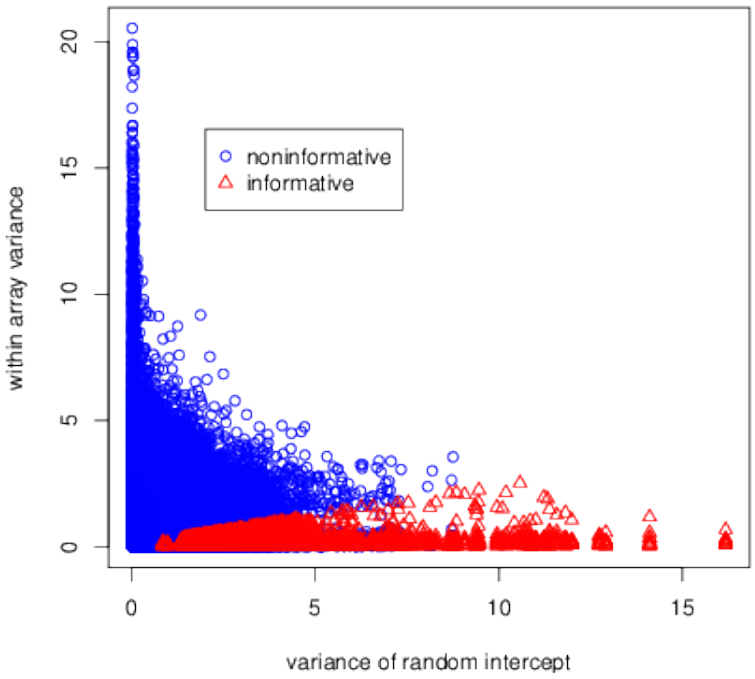


Figure 2: Variance of random intercept versus variance of measurement error. Bead types are declared informative based on thresholding the minimum ICC at a cutoff point of 80%: noninformative bead types (○) and informative bead types (△).

Acknowledgements

We acknowledge the support from IAP research network grant nr. P6/03 of the Belgian government (Belgian Science Policy), SymBioSys, the Katholieke Universiteit Leuven center of Excellence on Computational Systems Biology, (EF/05/007), and Bioframe of the institute for the Promotion of Innovation by Science and technology in Flanders (IWT: 060045/KUL-BIO-M$S-PLANT).

We are also grateful to Kim and Lin (2011) for making the `MLM.beadarray` function available.

Bibliography

S. Calza, W. Raffelsberger, A. Ploner, J. Sahel, T. Leveillard, and Y. Pawitan. Filtering genes to improve sensitivity in oligonucleotide microarray data analysis. *Nucleic Acids Research*, 35(16), 2007. [p171]

M. Dunning, N. B. Morais, A. Lynch, S. Tavaré, and M. Ritchie. Statistical issues in the analysis of Illumina data. *BMC Bioinformatics*, 9(1):85+, 2008. ISSN 1471-2105. doi: 10.1186/1471-2105-9-85. URL http://dx.doi.org/10.1186/1471-2105-9-85. [p173]

M. J. Dunning, M. L. Smith, M. E. Ritchie, and S. Tavaré. beadarray: R classes and methods for Illumina bead-based data. *Bioinformatics*, 23(16):2183–2184, Aug. 2007. ISSN 1367-4811. doi: 10.1093/bioinformatics/btm311. URL http://dx.doi.org/10.1093/bioinformatics/btm311. [p171, 173]

A. C. Forcheh, G. Verbeke, A. Kasim, D. Lin, Z. Shkedy, W. Talloen, H. W. Göhlmann, and L. Clement. Gene filtering in the analysis of Illumina microarray experiments. *Journal of Statistical Applications in Genetics and Molecular Biology*, 11(1), 2012. [p171, 172, 173, 176]

A. Kasim, D. Lin, S. Van Sanden, D.-A. Clevert, L. Bijnens, H. Göhlmann, D. Amaratunga, S. Hochreiter, Z. Shkedy, and W. Talloen. Informative or noninformative calls for gene expression: a latent variable approach. *Statistical applications in genetics and molecular biology*, 9(1), Jan. 2010. ISSN 1544-6115. doi: 10.2202/1544-6115.1460. URL http://dx.doi.org/10.2202/1544-6115.1460. [p171, 172]

R. S. Kim and J. Lin. Multi-level mixed effects models for bead arrays. *Bioinformatics*, 27:633–640, March 2011. ISSN 1367-4803. doi: http://dx.doi.org/10.1093/bioinformatics/btq708. URL http://dx.doi.org/10.1093/bioinformatics/btq708. [p172, 173, 179]

W. Talloen, D.-A. Clevert, S. Hochreiter, D. Amaratunga, L. Bijnens, S. Kass, and H. W. H. Göhlmann. I/NI-calls for the exclusion of non-informative genes: a highly effective filtering tool for microarray data. *Bioinformatics*, 23(21):btm478–2902, Oct. 2007. doi: 10.1093/bioinformatics/btm478. URL http://dx.doi.org/10.1093/bioinformatics/btm478. [p171, 172]

G. Verbeke and G. Molenberghs. *Linear mixed models for longitudinal data*. Springer Series in Statistics. Springer-Verlag, New-York, 2000. [p171]

Anyiawung Chiara Forcheh
Interuniversity Institute for Biostatistics and statistical Bioinformatics, Katholieke Universiteit Leuven, Kapucijnenvoer 35, Blok D, bus 7001, B3000 Leuven, Belgium,
and Universiteit Hasselt,
Belgium
forcheh@yahoo.com

Geert Verbeke and Lieven Clement
Interuniversity Institute for Biostatistics and statistical Bioinformatics, Katholieke Universiteit Leuven, Kapucijnenvoer 35, Blok D, bus 7001, B3000 Leuven,
Belgium and Universiteit Hasselt, Belgium
Geert.Verbeke@med.kuleuven.be
lieven.clement@med.kuleuven.be

Dan Lin and Ziv Shkedy
Interuniversity Institute for Biostatistics and statistical Bioinformatics, Universiteit Hasselt, Agoralaan 1, B3590 Diepenbeek, Belgium, and Katholieke Universiteit Leuven,
Belgium
dan.lin2@pfizer.com
ziv.shkedy@uhasselt.be

Adetayo Kasim
Wolfson Research Institute, Durham University,
Queen's Campus, University Boulevard, Thornaby,
Stockton-on-Tees,TS17 6BH, United Kingdom
a.s.kasim@durham.ac.uk

Willem Talloen and Hinrich W.H. Göhlmann
Janssen Pharmaceutica N.V.,
Beerse, Belgium
wtalloen@its.jnj.com
hinrich@goehlmann.info

Fast Pure R Implementation of GEE: Application of the Matrix Package

by Lee S. McDaniel, Nicholas C. Henderson and Paul J. Rathouz

Abstract Generalized estimating equation solvers in R only allow for a few pre-determined options for the link and variance functions. We provide a package, **geeM**, which is implemented entirely in R and allows for user specified link and variance functions. The sparse matrix representations provided in the **Matrix** package enable a fast implementation. To gain speed, we make use of analytic inverses of the working correlation when possible and a trick to find quick numeric inverses when an analytic inverse is not available. Through three examples, we demonstrate the speed of **geeM**, which is not much worse than C implementations like **geepack** and **gee** on small data sets and faster on large data sets.

Introduction

An extension of generalized linear models and quasilikelihood (McCullagh and Nelder, 1986), generalized estimating equations (GEEs; Liang and Zeger, 1986) are a useful method for analyzing clustered (or longitudinal) data which may be correlated within cluster but are independent between clusters.

There are two packages on CRAN which will solve GEEs and produce standard errors in R: **geepack** (Højsgaard et al., 2006) and **gee** (Carey. et al., 2012). **gee** provides the basic functionality necessary for fitting GEEs, while **geepack** also allows for regression models for both the scale and correlation parameters, as described in Yan and Fine (2004). Both of these packages rely heavily on a computational engine implemented in C and called by the R function wrapper. They use the `family` object as input and hard code each of the link and variance function options within the C code. Whereas this configuration yields computational speed, it has two important drawbacks: it does not allow the user to create and specify his own link and variance functions, and it is not easy for a programmer with only modest programming skills to alter the code for methodological developments related to or extending GEE.

To see why the user may need to define link and variance functions, consider the methodology presented in Schildcrout and Rathouz (2010). The authors look at binary response data in an outcome dependent sampling situation. Because the response is binary, the authors are able to use the offset option in standard GEE software to correct the bias from the sampling scheme. However, if the response takes some other form (e.g. counts or a continuous response), then we would need to define link and variance functions to correct this bias. These link and variance functions involve integrals that need to be computed whenever the functions are evaluated at different points.

Our new package, called **geeM** (the M is to emphasize the use of the **Matrix** package; Bates and Maechler, 2013) is coded completely in R, which grants the user the flexibility to define link and variance functions as part of the model specification (McDaniel and Henderson, 2013). It also allows for easy modification of the code by method developers. The main function in the package, `geem`, takes a list that includes the link function, variance function, inverse link function, and the derivative of the inverse link function, as an argument. These functions must all be vectorized.

In spite of this, we are still able to solve the estimating equations quickly for many problems. To accomplish this, the package relies heavily on the recently developed **Matrix** package for speed. We especially exploit the sparse matrix representation objects contained therein, which allow for efficient operations on matrices in which most elements are zero. Besides introducing our package, a key aim of this article is to demonstrate in the context of a well-known statistical algorithm the power, speed and flexibility of the **Matrix** package.

We first give the details of the implementation, and then describe some limitations of the package. Finally, we show through examples and simulations the relative speed of the new R package. Along the way, we point out innovative uses of **Matrix** package functions.

GEE solutions

To set up the problem, consider Y_i, a multivariate n_i-vector of responses $Y_i = (Y_{i1}, \ldots, Y_{it}, \ldots, Y_{in_i})$ with mean vector $\mu_i = (\mu_{i1}, \ldots, \mu_{it}, \ldots, \mu_{in_i})$. Denote $E(Y_{it}) = \mu_{it}$. Let $X_i = (x_{i1}, \ldots, x_{it}, \ldots, x_{in_i})'$ be a $n_i \times p$ design matrix of covariate vectors corresponding to each observation for the ith cluster.

In the typical specification, the expectations are related to the mean by the monotone link function,

$$g(\mu_{it}) = \eta_{it} = x_{it}^{\top}\beta\,,$$

and the variances are a function of the mean

$$\text{var}(Y_{it}) = \phi a_{it} = \phi a(\mu_{it})\,,$$

where ϕ is a dispersion parameter. Let A_i be a diagonal matrix with kth diagonal element equal to $a(\mu_{ik})$. Next we define

$$V_i = \phi A_i^{\frac{1}{2}} R_i(\alpha) A_i^{\frac{1}{2}}\,, \tag{1}$$

where $R_i(\alpha)$ is the working correlation matrix for the ith cluster, parameterized by α, which may be a vector.

Using these conventions, Liang and Zeger (1986) describe a modified Fisher scoring algorithm to estimate regression coefficients β. The generalized estimating equations are given by

$$\sum_{i=1}^{K} D_i^{\top} V_i^{-1} S_i = 0\,, \tag{2}$$

where $D_i = A_i \Delta_i X_i$ and Δ_i is a diagonal matrix with kth entry equal to $\frac{dg^{-1}(\eta_{ik})}{d\eta_{ik}}$. Finally, $S_i = Y_i - \mu_i$.

Instead of using a summation, we can instead put all of this in matrix form. Let $N = \sum_{i=1}^{K} n_i$, X be a $N \times p$ matrix with the X_i matrices stacked on top of one another, Δ and A be $N \times N$ diagonal matrices with the appropriate entries from Δ_i and A_i on the diagonal, and S be a N-vector generated by stacking up the S_i's. Finally, $R(\alpha)$ is a $N \times N$ block diagonal matrix with the individual $R_i(\alpha)$ matrices as blocks. With these definitions, our GEE representation is

$$X^{\top} \Delta A \left(A^{\frac{1}{2}} R(\alpha) A^{\frac{1}{2}} \right)^{-1} S = 0. \tag{3}$$

Implementation details

The **geeM** package is coded entirely in R. This allows for easier modification by methodologists considering variants and extensions of GEE, as well as increased flexibility for some specific analyses.

Uses of the Matrix package

Creating large matrices to solve the estimating equations leads to large sparse matrices. The **Matrix** package allows us to easily exploit this sparsity with its sparse matrix representations, which store only the non-zero elements of a matrix. Without the memory savings from these representations, even moderately sized problems would lead to huge memory demands.

Further, we benefit from an impressive gain in the speed of operating on these large sparse matrices. Multiplication is the largest burden computationally, and the speed gains in very large, very sparse matrices are immense. This is accomplished by a separately defined `%*%` method for objects in the "`sparseMatrix`" class. When multiplication is of the form $B^{\top}C$ or $B^{\top}B$ we can take advantage of the `crossprod` function.

In the modified Fisher scoring algorithm, we have many opportunities to take advantage of the "`sparseMatrix`" class. Two diagonal matrices are used, A and Δ, which we create using the `Diagonal` function. We also have the working correlation matrix, $R(\alpha)$, which is a sparse symmetric block diagonal matrix because observations within clusters may be correlated but observations between clusters are independent.

Additionally, the Fisher scoring algorithm requires that we compute the inverse of the working correlation matrix, $R(\alpha)$. Because we never need $R(\alpha)$ directly, we only compute $R(\alpha)^{-1}$ and build it directly for any given α. For that we use the `sparseMatrix` function, to which we supply three vectors of the same length: an integer vector `i` of row positions of non-zero entries, an integer vector `j` of column positions of non-zero entries, and a vector `x` of entries.

Analytic inverses

For three common correlation structures, independence, exchangeable, and AR-1, analytic inverses are available. In these cases we use these analytic inverses to create $R(\alpha)^{-1}$.

In the AR-1 case, we are able to quickly construct the inverse correlation matrix as a linear combination of three basic matrices, with coefficients depending on the value of α (Sutradhar and Kumar, 2003). The inverse can be constructed as

$$R(\alpha)^{-1} = \frac{1}{1-\alpha^2}\left\{L - \alpha M + \left(1+\alpha^2\right) N\right\},$$

where L, M, and N are all block diagonals, the blocks of which take the forms

$$L_i = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix},$$

$$M_i = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix},$$

and

$$N_i = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

These basic matrices are constructed only once at the beginning of the code and saved for later use.

The exchangeable correlation matrix is a little more complicated because the entries in the inverse depend on the cluster size (Sutradhar and Kumar, 2003). The analytic inverse can be computed as

$$R_i(\alpha)^{-1} = (a-b)I_{n_i} + bJ_{n_i},$$

where

$$a = \frac{1+(n_i-2)\alpha}{(1-\alpha)\{1+(n_i-1)\alpha\}},$$

$$b = -\frac{\alpha}{(1-\alpha)\{1+(n_i-1)\alpha\}},$$

I_{n_i} is the $n_i \times n_i$ identity matrix, and J_{n_i} is an $n_i \times n_i$ matrix of ones. In the implementation, we compute a vector with every entry in the correlation matrix (a vector of length $\sum_{i=1}^{K} n_i^2$) and create the new matrix with the `sparseMatrix` function.

Numeric inverses

For all other correlation structures, we use the `solve` command in R. To describe the method, let us suppose that we have k different cluster sizes, $n_1, \ldots, n_k$, in descending order.

We first construct the working correlation matrix for all clusters of size n_1, then take the upper left corner of this matrix to construct correlation matrices of appropriate size for clusters of size $n_2, \ldots, n_k$. We then calculate the inverses of each of these k matrices. Finally, we construct the block diagonal working correlation matrix for the whole sample using the inverses of the appropriate size.

This technique exploits the fact that the inverse of a block diagonal matrix is the inverse of the blocks, allowing us to reduce the number of invocations of the `solve` function to k, the number of unique cluster sizes.

Using this technique, we have support for M-dependence (M should be specified by the user), unstructured correlation, fixed correlation, and user-defined correlation structures. Under M-dependence, two observations, x_{it} and $x_{it'}$ have correlation $\alpha_{|t-t'|}$ if $|t-t'| \leq M$ and 0 otherwise. Under unstructured correlation, the correlation between x_{it} and $x_{it'}$ is assumed to be $\alpha_{tt'}$. The fixed correlation option requires that the user provides a correlation matrix giving the correlations between all observations in the largest cluster. For a user-defined correlation structure, the function accepts a square matrix of the size of the largest cluster which defines the entries in the correlation matrix assumed to be the same.

Note that in all correlation structures, if any variables are missing, then those rows are dropped. So, if rows have missing data, or a given subject has fewer than the maximum number of observations, then the leading principal submatrices of the appropriate dimensions are used. This may not preserve the desired correlation structure. The desired structure can be recovered by including rows with weights of zero for those time points with missing data.

Estimation of correlation parameters

For all supported correlation structures, we use simple moment estimators. These are the exact same estimators used by SAS proc genmod (SAS Institute Inc., 2010) and are estimated solely from the Pearson residuals.

Missing values

The `weights` argument can be used to handle missing values in the response by creating a new row for the missing observation and assigning a weight of 0. This will maintain the desired correlation structure and is equivalent to how SAS proc genmod handles missing data. Observations with a weight of 0 do not contribute to estimation of the coefficients or the variance and only observations with a positive weight are used to calculate the correlation parameters. Any observation with an `NA` in the response or predictors is automatically assigned a weight of 0.

Limitations

The **geeM** package calculates the inverse correlation matrix directly, possibly resulting in a less stable algorithm than **gee** and **geepack** which solve systems of equations instead of inverting directly. Therefore it is possible that the **geeM** package may have problems with numerical stability on some problems, especially for correlation structures for which inverses are calculated numerically. We have not seen any such issues in testing at this point.

Another issue has to do with how the calculations are completed. In testing the largest dataset (presented later as the birth outcomes data) we found that **geeM** was more likely to have memory problems than **gee** or **geepack**. This is probably because **geeM** creates large sparse matrices for computing its estimates as opposed to the technique in Liang and Zeger (1986), which involves summing over many more small dense matrices.

Speed comparisons

All speed comparisons were run on a computer with an Intel Core 2 Duo 3.0 GHz processor and 4 GB of RAM. Windows 7 was the operating system and R version 2.15.1 was used.

For these comparisons, we fit the same model to the same data set multiple times in order to gain some stability in the estimates of run time. The packages **geepack** and **gee** test convergence differently than **geeM**, so the tolerance used for **geeM** was chosen to be no less strict than the other two. Because **geeM** assesses convergence relative to the magnitude of parameters, this requires some knowledge of the values of the fitted parameters.

Ohio data set

geepack contains a data set on the wheezing status of children in Ohio. The data are 537 clusters each with 4 observations. The response (`resp`) is a binary variable with 1 corresponding to wheezing and 0 not. We fit the model `resp ~ age + smoke + age:smoke`, where `age` is the age of the child and `smoke` is an indicator of maternal smoking at the first year of the study. In this case we use a logit link and the $\mu(1-\mu)$ variance function.

Table 1 shows the speed results under four different correlation structures. The average speed (Avg.) is calculated based on fitting the same data 5 times for each method and the relative speed (Rel.) is the average speed of each method divided by the average speed of the fastest method.

This is a relatively small data set, so all solvers perform very quickly in this case. The longest average time for any of them is 0.20 seconds, so speed does not appear to be too much of an issue here.

	Independence		AR-1		Exchangeable		Unstructured	
	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.
geeM	0.11	1.83	0.11	1.00	0.11	1.33	0.16	1.11
gee	0.07	1.17	0.12	1.07	0.08	1.00	0.15	1.00
geepack	0.06	1.00	0.18	1.53	0.12	1.48	0.20	1.34

Table 1: Speed comparisons for `ohio` data set.

Simulated count data

For a slightly larger data set, we simulated count data from an overdispersed Poisson distribution. There are 10,000 clusters with 5 observations in each cluster. The formula we use is `count ~ time`. Here we use the log-linear link and the identity variance function.

Table 2 shows the speed results under four different working correlation structures. The average speed (Avg.) is calculated based on fitting the same data 5 times for each method and the relative speed (Rel.) is the average speed of each method divided by the average speed of the fastest method.

	Independence		AR-1		Exchangeable		Unstructured	
	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.
geeM	1.36	1.00	1.56	1.00	1.58	1.00	2.30	1.00
gee	2.16	1.59	4.06	2.60	2.45	1.55	3.26	1.42
geepack	1.55	1.14	3.27	2.10	2.95	1.86	5.64	2.46

Table 2: Speed comparisons for the simulated count data set.

Birth outcomes data

Finally, we move to a large data set. These data contains the information on birth outcomes from Abrevaya (2006). We use gestational age as the continuous response, and mother's age as the only covariate. We use the Gaussian family for the link and variance functions.

The data contain 141,929 clusters of size 2 or 3 each. This results in a total of 296,218 observations. It is worth noting that for a data set this size, all other programs except R needed to be shut down. If too many other programs are open, **geeM** may take a much longer time or even run out of memory.

Table 3 shows the speed results under four different working correlation structures. The average speed (Avg.) is calculated based on fitting the data 5 times for each method and the relative speed (Rel.) is the average speed of each method divided by the average speed of the fastest method.

	Independence		AR-1		Exchangeable		Unstructured	
	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.	Avg. (s)	Rel.
geeM	5.28	1.00	9.87	1.00	7.33	1.00	8.18	1.00
gee	10.02	1.90	29.39	2.98	20.09	2.74	21.49	2.63
geepack	9.66	1.83	23.67	2.40	22.59	3.08	25.67	3.14

Table 3: Speed comparisons for the birth outcomes data.

Illustrative use

Before concluding, we provide examples of how to use the `geem` function. Here we fit the `ohio` data with different link and variance functions. In this case, we use a skewed logit link function, as described in Prentice (1976), which allows us to remove the symmetry constraint imposed by the logit and probit link functions. According to Prentice, this type of skewed logit link may provide a better fit to dose response data. The inverse link function takes the form

$$p = \left(\frac{e^{x^\top \beta}}{1 + e^{x^\top \beta}} \right)^k,$$

where k is known. When $k = 1$ we have the logistic model, when $k < 1$ the distribution is negatively skewed, and when $k > 1$ the distribution is positively skewed. In the example, we let $k = 2$ and choose $\mu(1-\mu)$ as the variance function. When defining these four functions, it is essential that they all be vectorized, that is, they accept a vector argument and return a vector of the same length.

```
k <- 2
linkfun <- function(p) {
  log((p^(1/k))/(1 - p^(1/k)))
}
variance <- function(p) {
  p * (1-p)
}
linkinv <- function(eta) {
  (exp(eta)/(1 + exp(eta)))^k
}
mu.eta <- function(eta) {
  k * (exp(eta))^(k-1) / (1 + exp(eta))^(k+1)
}
FunList <- list(linkfun, variance, linkinv, mu.eta)
geem(resp ~ age + smoke + age:smoke, id=id, data = ohio, family = FunList,
 corstr = "unstructured")
```

If we just want to use the binomial family to fit the ohio data, then the call is:

```
geem(resp ~ age + smoke + age:smoke, id=id, data = ohio, family = binomial,
 corstr = "unstructured")
```

Conclusion

This paper describes a pure R implementation of a generalized estimating equation solver relying heavily on the **Matrix** package. Coding in R allows for the use of user-defined link and variance functions, giving a useful kind of flexibility.

The **Matrix** package is what enables a fast implementation. Sparse matrix representations give ways to efficiently build the large, sparse matrices needed to solve the GEEs without using too much memory. Operations on these sparse matrices are also much faster.

Much of the work in fitting GEEs comes from matrix inversion or solving a system of equations. For three correlation structures, we are able to analytically invert the largest matrix. For the rest of the correlation structures, we rely on inversion of a relatively small number of blocks to find the inverse.

Unfortunately, this technique of gaining speed in matrix inversion can cause problems. Because we invert matrices instead of solving systems, it is possible that the function may be less numerically stable.

Finally, we show that, in some cases, the **geeM** package is faster than **gee** or **geepack**. In other cases it is not much slower, with **geeM** tending to run faster on larger data sets.

Acknowledgements

McDaniel's and Henderson's efforts were funded by NIGMS grant 5T32GM074904. Rathouz's effort was funded by NIH grant R01 HL094786.

Bibliography

J. Abrevaya. Estimating the effect of smoking on birth outcomes using a matched panel data approach. *Journal of Applied Econometrics*, 21(4):489–519, 2006. [p185]

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2013. URL http://CRAN.R-project.org/package=Matrix. R package version 1.0-12. [p181]

V. J. Carey., T. Lumley, and B. Ripley. *gee: Generalized Estimation Equation solver*, 2012. URL http://CRAN.R-project.org/package=gee. R package version 4.13-18. [p181]

S. Højsgaard, U. Halekoh, and J. Yan. The R package geepack for generalized estimating equations. *Journal of Statistical Software*, 15(2):1–11, 2006. [p181]

K. Liang and S. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1): 13–22, 1986. [p181, 182, 184]

P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman and Hall, second edition, 1986. [p181]

L. McDaniel and N. Henderson. *geeM: Fit Generalized Estimating Equations*, 2013. URL http://CRAN.R-project.org/package=geeM. R package version 0.06. [p181]

R. Prentice. A generalization of the probit and logit methods for dose response curves. *Biometrics*, 32 (4):761–768, 1976. [p185]

SAS Institute Inc. *SAS/STAT Software, Version 9.22*. Cary, NC, 2010. URL http://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm#statug_genmod_sect049.htm. [p184]

J. Schildcrout and P. Rathouz. Longitudinal studies of binary response data following case-control and stratified case-control sampling: design and analysis. *Biometrics*, 66(2):365–373, 2010. [p181]

B. Sutradhar and P. Kumar. The inversion of correlation matrix for MA(1) process. *Applied Mathematics Letters*, 16(3):317–321, 2003. [p183]

J. Yan and J. P. Fine. Estimating equations for association structures. *Statistics in Medicine*, 23(6):859–880, 2004. [p181]

Lee S. McDaniel
University of Wisconsin-Madison
1300 University Ave.
Madison, WI 53706 USA
mcdaniel@stat.wisc.edu

Nicholas C. Henderson
University of Wisconsin-Madison
1300 University Ave.
Madison, WI 53706 USA
nhenders@stat.wisc.edu

Paul J. Rathouz
University of Wisconsin-Madison
1300 University Ave.
Madison, WI 53706 USA
rathouz@biostat.wisc.edu

RcmdrPlugin.temis, a Graphical Integrated Text Mining Solution in R

by Milan Bouchet-Valat and Gilles Bastin

Abstract We present the package **RcmdrPlugin.temis**, a graphical user interface for user-friendly text mining in R. Built as a plug-in to the R Commander provided by the **Rcmdr** package, it brings together several existing packages and provides new features streamlining the process of importing, managing and analyzing a corpus, in addition to saving results and plots to a report file. Beyond common file formats, automated import of corpora from the Dow Jones Factiva content provider and Twitter is supported. Featured analyses include vocabulary and dissimilarity tables, terms frequencies, terms specific of levels of a variable, term co-occurrences, time series, correspondence analysis and hierarchical clustering.

Introduction

Text mining applications have become quite popular in the social sciences – in particular in sociology and political science – fostered by the availability of integrated graphical user interfaces (GUIs) that render typical analyses possible at a reasonably low learning cost. R has been technically well positioned in the area of text mining for several years, thanks to the **tm** package (Feinerer et al., 2008; Feinerer and Hornik, 2013) and to its rich ecosystem of packages dedicated to advanced text mining operations[1], but also to general purpose packages whose power can be leveraged for this particular application. Yet, the power and the flexibility of this environment present the downside that non-technical users can feel lost, especially when they are used to GUIs that do not provide them – to say the least – with so many possibilities.

In this article, we present **RcmdrPlugin.temis** ("R.TeMiS", for R Text Mining Solution, in short) version 0.6.1 (Bouchet-Valat and Bastin, 2013), a new package that intends to fill this gap by providing users with a GUI, in the form of a menu augmenting the R Commander (provided by the **Rcmdr** package, *cf.* Fox, 2005). We hope to provide an integrated and easy to use graphical solution that does not constrain the user into a specific path of analysis, as an alternative to proprietary software packages that currently dominate the market, often with a closed-source conception of what users should (and can) do, and what they should not.

Following the spirit of free software and scientific research, we intend to empower users and not lock them in a jointly technical and theoretical, rigorously defined paradigm. Though, even if we are open to different approaches of text mining, the current features that we demonstrate below are largely based on a long-established tradition of exploratory data analysis; the interested reader will find a detailed description of these techniques in e.g. Lebart et al. (1998).

Design choices

The fundamental choice behind the design of **RcmdrPlugin.temis** is that it is fully integrated into – and actually written in – R. Following the general principle adopted by the R Commander, every action initiated by the user via the GUI generates R code which is shown to the user while it is executed (Figure 1), and which can be saved and run again later. This makes it possible to manually edit the code, be it to customize a function call or to reuse objects to extend the analysis without starting from scratch; this also ensures the reproducibility of the results, a major issue in scientific research.

Thus, **RcmdrPlugin.temis** differs from other text mining GUIs, which do not generate source code, and provide little space for extending the set of possible analyses. In particular, two free software text mining tools offer a relatively similar set of features: Iramuteq (Ratinaud, 2013) and TXM (Heiden, 2010)[2]. These two projects use R as a back end for some analyses, but are written as standalone applications completely separated from the R session; this means that the power of R as a general purpose statistical environment is not directly accessible to the user.

The decision of writing our package as an R Commander plug-in also presents several other advantages. First, all the features offered by **Rcmdr** are still available to the user: in particular, menus allowing to import and recode data and to export plots can be very useful for text mining work.

[1] See the "Natural Language Processing" CRAN task view at http://CRAN.R-project.org/view=NaturalLanguageProcessing for a list of packages applying to this field.

[2] See respectively http://www.iramuteq.org and http://textometrie.ens-lyon.fr/?lang=en. Interestingly, all of the three projects are developed by French researchers.

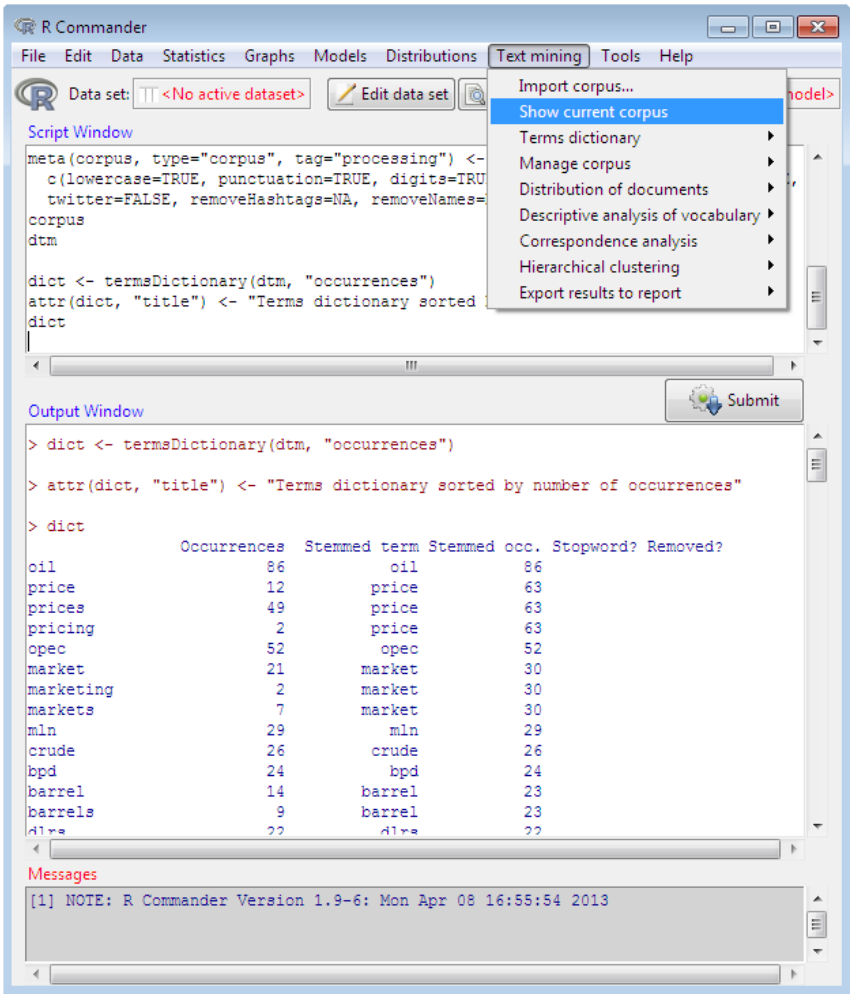


Figure 1: Main R Commander window showing text mining menu and top of the terms dictionary sorted by number of occurrences.

Second, **Rcmdr** runs without too much tweaking on all platforms, since the toolkit it relies on, Tcl/Tk, is built-in standard in R via the **tcltk** package (see the next section for instructions on how to install **RcmdrPlugin.temis**). The downside of this choice is of course that the resulting interface is not fully geared towards text mining, contrary to what a dedicated application would have made possible.

Other general purpose R GUIs such as RKWard (Rödiger et al., 2012) and Deducer (Fellows, 2012) could have been used instead of the R Commander. But at the time the project was started, no Mac OS X port of the former existed, and the latter was not even released yet (for a discussion of the different R GUIs see Valero-Mora and Ledesma, 2012, and the individual articles of that special issue). The choice of the hosting GUI for a text mining plug-in is in the end closely tied to the question of the best R GUI: our choice has been to rely on the most mature one[3].

Under the hood, **RcmdrPlugin.temis** puts together state-of-the-art R packages covering various types of applications and makes them easily accessible to perform text mining tasks. At the core of the text mining operation is the **tm** package, which provides the framework to import corpora from various sources, process texts and extract their vocabulary in the form of a *document-term matrix*. All other packages on which **RcmdrPlugin.temis** depends are plugged into this framework to provide specific features. We illustrate them below.

Installing and launching the package

Like any CRAN package, **RcmdrPlugin.temis** and its mandatory dependencies can be installed using the R command

[3]The GUI itself represents only a share of the work required to create an integrated text mining solution: this means that plug-ins for other general purpose GUIs could be written without too much effort in the future by adapting **RcmdrPlugin.temis**'s code base.

```
install.packages("RcmdrPlugin.temis")
```

or from graphical menus provided by the R GUI you are using.

In case of problems, more details concerning that platform, as well as Windows and Linux, can be found on the R Commander web page: http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/installation-notes.html.

Once installed, the package can be be loaded with the command:

```
library("RcmdrPlugin.temis")
```

(or, again, using menus).

Importing and managing corpora

The first task that **RcmdrPlugin.temis** intends to allow is the seamless importation of a corpus – an often neglected task which can prove quite time-consuming in typical research work. The program will ask to automatically install the packages needed to import corpora from the chosen source at the first attempt to do so[4]: this applies to all sources except plain text and comma/tab-separated values (CSV/TSV) files. Four types of sources are currently supported:

- A series of plain text files (typically .txt) contained in a directory.
- A spreadsheet-like file (.csv, .ods, .xls, ...) with one line per document (answers to an open-ended question, typically). The first column must contain the text to analyze; the remaining columns can provide information about the document and are automatically imported as variables. Open Document Spreadsheet files (.ods) are loaded using the **ROpenOffice** package (Temple Lang, 2011)[5], and Microsoft Excel files (.xls, .xlsx) are imported using the **RODBC** package (Ripley and Lapsley, 2012)[6].
- One or several .xml or .html files exported from the Dow Jones Factiva content provider, using the dedicated **tm.plugin.factiva** package (Bouchet-Valat, 2013b); using this portal, often available via academic subscriptions, streamlines the process of creating a corpus by allowing the mass importation of press articles from many newspapers, together with meta-data information like origin, date, author, topics and geographic coverage.
- A Twitter search (thanks to the **twitteR** package, *cf.* Gentry, 2013) on hash tags, authors or full text, or more complex criteria supported by the Twitter API.

By default, plain text and CSV/TSV files are assumed to use the native system encoding: this is usually UTF-8 (Unicode) on Mac OS X and Linux, and varies depending on the locale on Windows. An indication that this default choice is not appropriate is the presence of incorrect accentuated characters in the corpus, but it may also happen that the import fails, or does not detect documents and variables. In that case, you will need to specify the encoding of the file by choosing it in the drop-down list provided by the import dialog (Figure 2). If such problems persist, opening the relevant files in a text editor or spreadsheet application and saving them in a known encoding is a reliable solution[7].

The import dialog also offers text processing options. First, texts can be split into smaller chunks defined as a number of adjacent paragraphs, which is most useful for relatively long texts, for example press articles imported from a Factiva source; in that case, each chunk will be considered as a document in subsequent analyses, and a meta-data variable will identify the original text from which the chunk originates. Second, texts can be processed to make them more suitable for computer analysis: conversion to lower case, punctuation removal, stopwords removal, and finally stemming can be carried out. Stopwords removal is disabled by default, since we consider that it should only be performed based on a conscious decision made by the researcher: in many cases, at least some stopwords are of interest for the analysis.

[4] This operation requires a working Internet connection.

[5] This package will only be automatically installed by the GUI on Linux at the time of writing, since no binary packages are available. Users familiar with building R packages from source can install the package manually, but this requires tools which are typically not installed by most users on Windows and Mac OS X.

[6] This package will only be automatically installed by the GUI on Windows, since **RODBC** relies on an Excel ODBC driver which is installed by default on Windows but not on other platforms. Mac OS X users can install such a driver and the package manually. New solutions might be investigated in the future.

[7] For example, OpenOffice/LibreOffice Writer and Calc applications offer this option – called "charset" – when "Edit filter options" is checked in the save dialog.

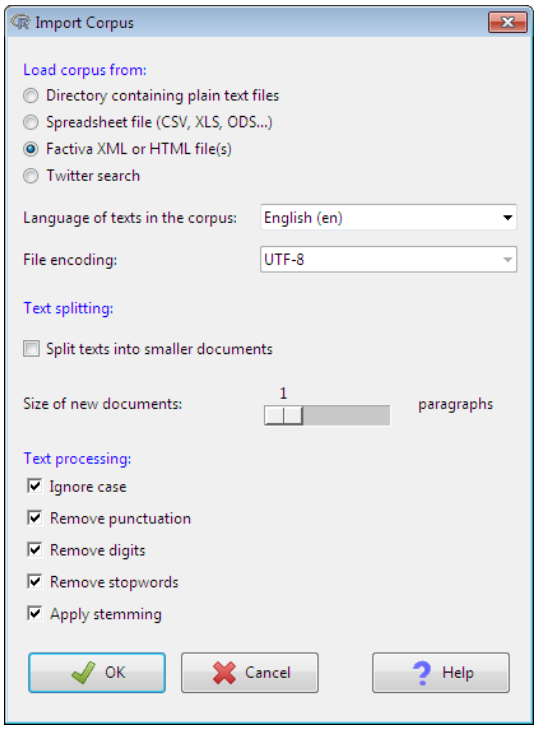


Figure 2: The import dialog with the settings used in the presented example.

Stemming is performed using the **SnowballC** package (Bouchet-Valat, 2013a), which relies on the C libstemmer library provided by the Snowball project initiated by Martin Porter[8]. This processing option is most interesting for short texts because small grammatical variations can dramatically reduce the number of (exact) co-occurrences between two texts, thus hindering later analyses of between-document relationships. After this step, the document-term matrix is created, which will be the basis for most later work; a summary of the matrix is printed on the screen.

Once the corpus has been imported, additional meta-data variables can be loaded from the active data set, be it imported from an external file or filled from R itself, using the '`Manage corpus` → `Set corpus variables`' menu item. The variables will then be made available for all analyses we present below.

The dictionary of terms that are present as columns of the document-term matrix and will be the basis of analyses can be altered in two ways:

- If specific terms are known (or found out later) to disturb the analyses – as commonly happens with meta-vocabulary like "read" and "page" in press articles – they can be excluded.
- If only some terms known in advance are of interest, the matrix can be restricted to them.

Finally, documents can be managed in the same way, by only keeping those containing (or *not* containing) some terms, or those corresponding to a given level of a variable. As said earlier, documents can refer to text chunks if this option was checked during the import, in which case this feature can be used to create a thematic corpus from large and heterogeneous original texts, thanks to a simple selection of chunks according to a set of terms, or to hierarchical clustering. In any case, the original corpus can be saved and restored at will.

For demonstration purposes, we use in the rest of this presentation a small corpus of 19 documents dealing with crude oil which appeared in the Reuters newswire in 1987, and is included in the classic Reuters 21578 data set (already used in **tm** examples). This small set of articles provides a good illustration of the typical analysis of a press corpus[9]. The corpus is shipped with the **tm.plugin.factiva**

[8]See `http://snowball.tartarus.org`. At the time of writing, supported languages are Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Romanian, Russian, Spanish, Swedish and Turkish.

[9]Even though most techniques make more sense applied to large corpora, this small set of texts is sufficient to illustrate all the steps involved in the analysis of a corpus of any size, with the advantage of easy reproducibility and low computing requirements.

package in the Factiva XML format with its meta-data; the location of this file on your system can be found using the following command provided you have installed the package:

```
system.file("texts", "reut21578-factiva.xml", package = "tm.plugin.factiva")
```

You can also download it directly from:
http://r-temis.r-forge.r-project.org/texts/reuters21578-factiva.xml.

To import the file as a corpus, use the 'Text mining → Import corpus...' menu, select 'Factiva XML or HTML file(s)' for source, choose "English (en)" as the language of the corpus, and enable stopwords removal by checking the corresponding box; we leave other options to their default values (Figure 2). Then click 'OK': dialogs will allow you to select the file ('reut21578-factiva.xml') and the variables to retain (we will use "Date", but you may also retain other variables), and the import phase will be run. You can then check the raw contents (*i.e.* not affected by processing operations) of the corpus using the 'Show current corpus' menu item, and see the corpus' words, their stemmed form and their number of occurrences using the 'Terms dictionary → Sorted by number of occurrences' menu (Figure 1).

Elementary statistics

Because **RcmdrPlugin.temis** aims at providing users with a complete text mining solution, it provides dialog boxes to compute simple statistics not directly related to lexical analysis, but that we consider useful to explore the composition of a corpus. The 'Distribution of documents' sub-menu allows creating one- and two-way tables from meta-data variables, and optionally plots the result. This sub-menu also makes it possible to plot time series representing the number of documents over time, using a single curve, or one curve for each level of a variable; as an option, the rolling mean can be computed over a configurable time window. This feature is based on packages **zoo** (Zeileis and Grothendieck, 2005) for computations and **lattice** (Sarkar, 2008) for plotting, which means the generated code can easily be extended by users for custom representations if needed. It is especially useful for corpora imported directly from Factiva or Twitter, which contain date and time information that can prove of high value for the study of the timing of media cycles.

The 'Descriptive analysis of vocabulary' menu introduces us to tasks more oriented towards text analysis *per se*. As a first preview of our corpus, we can print the most frequent terms and their number of occurrences, for the whole corpus as well as for each level of a variable.

A slightly more sophisticated information is provided by the 'Terms specific of levels...' dialog: this feature reports terms whose observed frequency in each level is either too high or too low compared to what would be expected given the documents' lengths and the global distribution of terms in the corpus. p-values based on an hypergeometric distribution give the probability of observing such an extreme number of occurrences in the level under the independence hypothesis; the sign of t-values can be used to identify positive (printed first) and negative (printed last) associations.

	% Term/Level	% Level/Term	Global %	Level	Global	t value	Prob.
post	2.0	67	0.58	8	12	3.3	0.0004
demand	1.3	71	0.34	5	7	2.7	0.0038
analyst	1.5	60	0.48	6	10	2.6	0.0051
product	1.8	50	0.68	7	14	2.4	0.0091
contract	1.0	67	0.29	4	6	2.2	0.0145
compani	1.3	56	0.43	5	9	2.1	0.0163
crude	2.5	38	1.26	10	26	2.1	0.0172
meet	1.8	44	0.77	7	16	2.0	0.0212
saudi	0.0	0	1.06	0	22	−2.4	0.0089
kuwait	0.0	0	0.92	0	19	−2.1	0.0170

Table 1: Terms specific of the first day with more than 5 occurrences in the corpus.

The printed output (*cf.* Table 1) also provides several descriptive statistics: the share of all occurrences of all terms in the level that is accounted for by the corresponding term ("% Term/Level"), the share of the occurrences of the term that happen in the level rather than elsewhere in the corpus ("% Level/Term"), the share of the term in all occurrences of all terms in the whole corpus ("Global %"), and finally the absolute number of occurrences of the term in the level and in the corpus.

In our example, choosing the "Date" variable, we can observe (Table 1, where terms with the lowest p-value and more than 5 occurrences are retained) that articles deal with oil markets on the

first day, while Middle East countries ("saudi", "kuwait") are not cited at all, contrary to other days, a difference from the corpus average which is significant at the 5% confidence level.

Other measures can be computed, both by document and by levels of a variable: vocabulary summary (number of terms, diversity and complexity); frequency and specificity of chosen terms; co-occurring terms (that is, terms that tend to appear in the same documents as chosen terms); dissimilarity table (Chi-squared distance between row profiles of the document-term matrix collapsed according to the levels of a variable).

Hierarchical clustering and correspondence analysis

The power of R and of its ecosystem is clearly visible when implementing more sophisticated techniques: from the document-term matrix created using **tm**, **RcmdrPlugin.temis** easily performs a hierarchical clustering of the documents using function 'hclust()' from the **base** package, and a correspondence analysis (CA) using the **ca** package (Nenadic and Greenacre, 2007). These two features are often used together in text mining software, but are relatively hard to run manually for beginners; the possibility to only plot a subset of points was also missing from all CA packages available in R, a feature particularly required for text mining where terms are too many to be all shown at the same time.

For both techniques, a dialog allows eliminating from the document-term matrix *sparse terms*, *i.e.* terms that appear in too few documents. Such terms are of little use for methods that intend to group a significant number of documents together based on term co-occurrences, and they can lead to system memory issues with large corpora. Here we use the value of 94%, which means that terms present in 6% of documents and less are not taken into account: this amounts to removing terms present in only one document, reducing the number of terms from 705 to 239 without much loss of information (since terms appearing in only one document provide no information on relations among documents, and can only help singling out outliers).

Hierarchical clustering can be run from the 'Hierarchical clustering → Run clustering...' menu; this command uses Ward's method based on a Chi-squared distance between documents[10].

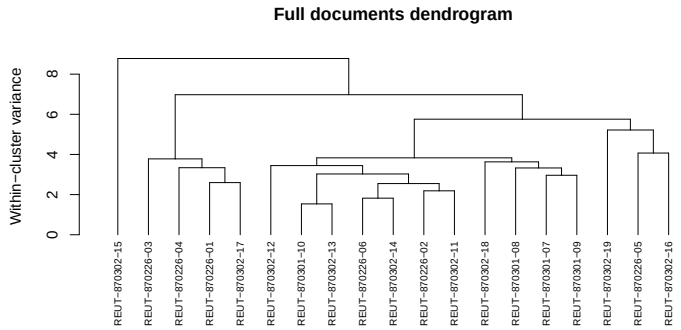


Figure 3: Dendrogram of documents resulting from hierarchical clustering.

The resulting dendrogram is automatically plotted (Figure 3), and its inspection prompts us to choose to create 5 clusters according to a scree test (the reduction in variance is relatively strong when going from 4 to 5 clusters, and slows down afterwards). If we change our mind later, the 'Create clusters...' menu command allows choosing another number of clusters without running the full computations again. As printed in a separate window, the resulting clusters contain 1, 4, 11, 1 and 2 documents, respectively; clusters 2 and 3 are more homogeneous (within-cluster variance of 3.8) than cluster 5, despite being the two largest, with respectively 21% and 58% of all documents versus 11%. Terms and documents specific of each cluster are also automatically printed. Specific documents are chosen according to their (as small as possible) Chi-squared distance to the centroid of the cluster, *i.e.* the mean term distribution of the cluster.

We will not detail here the interpretation of the clusters. Let us simply note that if one is interested in reading more associated terms, the 'Create clusters...' dialog will allow raising the total number of printed terms. Once identified, clusters can be used in all other analyses offered by

[10]Since this method tends to start with clusters identifying outliers, which are often of little interest, an alternative solution is also provided: instead of running the clustering procedure on the document-term matrix itself, one can choose to base the analysis on an approximation of the matrix obtained using a given number of axes of the CA (once it has been computed using the corresponding menu). This procedure reduces the noise in the data and allows for a strong relationship between clusters and CA axes. We do not retain this solution here.

RcmdrPlugin.temis: a meta-data variable called "Cluster" is automatically created. As an illustration, the '`Distribution of documents` → `Two-way table of variables...`' can now be used to cross the "Date" variable with the "Cluster" one, and see that cluster 3 gains momentum in March compared to February.

When dealing with a larger corpus with date or time information, the '`Temporal evolution...`' item available from the same sub-menu is of great interest to help visualize the variations of clusters over time. Further, the "Cluster" variable can be introduced in a CA as a supplementary variable, which can help identifying the meaning of clusters themselves: this is what we will now present.

But first, as the only article in cluster 1 appears to be very different from the rest of the corpus, we may want to exclude it before running the CA: indeed, if we retain it, we can notice that this article alone draws the opposition on several dimensions. We thus use the '`Manage corpus` → `Subset corpus` → `By variable...`' menu, choosing the newly created "Cluster" variable, and selecting only the levels 2, 3, 4 and 5 (keep the Ctrl key pressed to select several items in the box using the mouse).

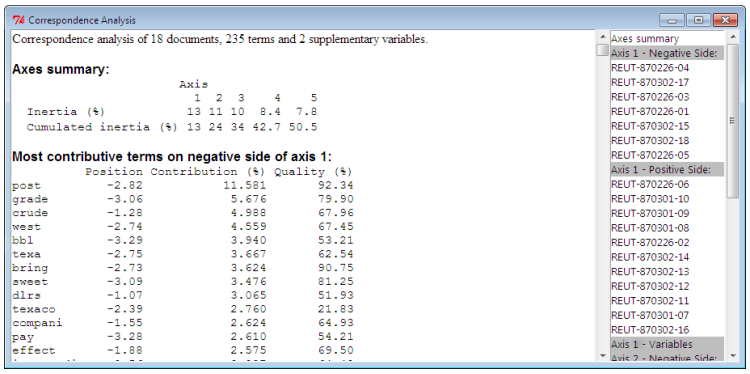


Figure 4: Information about the CA axes, and most contributive terms on the negative side of axis 1.

The CA can be run from '`Correspondence analysis` → `Run analysis...`': again, we exclude terms missing from more than 94% of documents. The number of dimensions to retain only affects the display of axes, and does not change the results: we can safely keep the default value. After running the CA, a dialog box opens automatically to show the results. It offers a few options that apply both to the text window that opens, similar as for the cluster analysis, and to the plot which is drawn at the same time. Among the most important options is the option which allows to select the number of terms and documents shown: as with clustering, only the most contributive items are shown; one can choose to show items most contributive to both axes taken together, or only to one of them. Here, we choose to print terms and documents most contributive to the first two axes.

The text window (Figure 4) presents axes one by one, first their negative, then their positive side, each with most contributive terms and documents, and finally the situation of supplementary (passive) variables on the axis. It thus constitutes a major help to interpreting the plotted plane (Figure 5).

By ticking the '`Variables`' check box in the '`Draw labels for:`' section, we can study the position of clusters on the CA plane by plotting them as supplementary levels: it is possible to select the "Cluster" variable so that other variables are not drawn (as previously, use the Ctrl key when clicking to select if you want to select several variables in the list).

Exporting results

Since **RcmdrPlugin.temis** aims at providing an integrated text mining solution, we could not leave our users at this stage of the analysis without providing them with a good way of exporting results. The last sub-menu is here precisely for that. After any of the operations described above, it allows copying the last printed table or last drawn plot to an output file in the HTML format, thanks to the **R2HTML** package (Lecoutre, 2003). With this command, plots are saved to a PNG file using a reasonably high resolution in the directory containing the HTML file, and inserted into it. The '`Draw plots in black and white`' menu command converts current and future plots to a mode suitable for printing and publications, using different line types to replace colors[11].

From the HTML file that is opened automatically in the default Web browser, users can copy tables to a word processor without loss of quality. Users looking for high resolution plots will happily use the richer exporting features of the R Commander ('`Graphs` → `Save graph to file`' menu) or of their standard R GUI, in particular to export graphs in vector formats like PDF, PostScript or SVG.

[11] CA plots, which are not drawn using **lattice**, do not benefit from this feature.

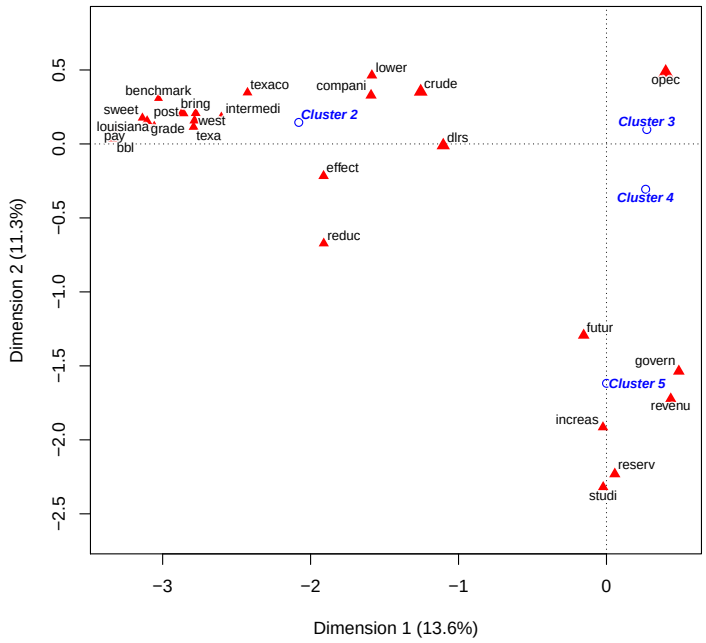


Figure 5: The first plane of the CA with most contributive terms and clusters as supplementary levels.

Given the low quality of graphics produced by most proprietary text mining software, we have no doubt that R's qualities in that area will be appreciated!

Conclusion

By putting together in a GUI the many existing pieces that already make possible advanced and flexible text mining analyses with R, we hope to fill the needs of new users, who are currently locked using proprietary solutions. We also hope to ease the life of more advanced users who can find in our package a straightforward way of running common tasks before adapting or extending the generated code.

Those looking for more detailed information than was exposed in this introduction about the exact operations run by the GUI are advised to consult **RcmdrPlugin.temis**'s documentation available under the 'Help' button in all dialog boxes, and to already cited introductions to the **tm** package, as well as online functions documentation.

RcmdrPlugin.temis does not impose on users any canonical model of text mining, but intends to provide users with all available techniques: we are open to suggestions and additions of new features that will help leverage the power of R's ecosystem.

Acknowledgments

The authors would like to thank the maintainers of the packages used by this GUI, in particular Ingo Feinerer, John Fox, Kurt Hornik and Duncan Temple Lang, who have been most helpful by their responses and the fixes they applied to their code to make it suit our needs when necessary, and Bénédicte Garnier and Élisabeth Morand (INED) for their useful remarks and harsh testing.

Bibliography

M. Bouchet-Valat. *SnowballC: Snowball stemmers based on the C libstemmer UTF-8 library*, 2013a. URL http://CRAN.R-project.org/package=SnowballC. R package version 0.5. [p191]

M. Bouchet-Valat. *tm.plugin.factiva: Import Articles from Factiva Using the tm Text Mining Framework*,

2013b. URL http://CRAN.R-project.org/package=tm.plugin.factiva. R package version 1.2. [p190]

M. Bouchet-Valat and G. Bastin. *RcmdrPlugin.temis: Graphical Integrated Text Mining Solution*, 2013. URL http://CRAN.R-project.org/package=RcmdrPlugin.temis. R package version 0.6.1. [p188]

I. Feinerer and K. Hornik. *tm: Text Mining Package*, 2013. URL http://CRAN.R-project.org/package=tm. R package version 0.5-8.3. [p188]

I. Feinerer, K. Hornik, and D. Meyer. Text mining infrastructure in R. *Journal of Statistical Software*, 25 (5):1–54, 2008. URL http://www.jstatsoft.org/v25/i05/. [p188]

I. Fellows. Deducer: A data analysis GUI for R. *Journal of Statistical Software*, 49(8):1–15, 2012. URL http://www.jstatsoft.org/v49/i08/. [p189]

J. Fox. The R Commander: A basic statistics graphical user interface to R. *Journal of Statistical Software*, 14(9):1–42, 2005. URL http://www.jstatsoft.org/v14/i09/. [p188]

J. Gentry. *twitteR: R Based Twitter Client*, 2013. URL http://CRAN.R-project.org/package=twitteR. R package version 1.1.0. [p190]

S. Heiden. The TXM platform: Building open-source textual analysis software compatible with the TEI encoding scheme. In R. Otoguro, K. Ishikawa, H. Umemoto, K. Yoshimoto, and Y. Harada, editors, *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, pages 389–398, Sendai, Japan, Nov. 2010. Institute for Digital Enhancement of Cognitive Development, Waseda University. URL http://halshs.archives-ouvertes.fr/halshs-00549764. [p188]

L. Lebart, A. Salem, and L. Berry. *Exploring Textual Data*. Kluwer Academic Press, Dordrecht/Boston, 1998. ISBN 0-7923-4840-0. [p188]

É. Lecoutre. The R2HTML package. *R News*, 3(3):33–36, 2003. URL http://CRAN.R-project.org/doc/Rnews/Rnews_2003-3.pdf. [p194]

O. Nenadic and M. Greenacre. Correspondence analysis in R, with two- and three-dimensional graphics: The ca package. *Journal of Statistical Software*, 20(3):1–13, 2007. URL http://www.jstatsoft.org/v20/i03/. [p193]

P. Ratinaud. *Iramuteq: Interface de R pour les Analyses Multidimensionnelles de Textes et de Questionnaires*, 2013. URL http://sourceforge.net/projects/iramuteq/files/iramuteq-0.6-alpha3/. Version 0.6-alpha3. [p188]

B. Ripley and M. Lapsley. *RODBC: ODBC Database Access*, 2012. URL http://CRAN.R-project.org/package=RODBC. R package version 1.3-6. [p190]

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/. [p189]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. [p192]

D. Temple Lang. *ROpenOffice: Basic Reading of Open Office Spreadsheets and Workbooks*, 2011. URL http://www.omegahat.org/ROpenOffice/. R package version 0.4-0. [p190]

P. M. Valero-Mora and R. Ledesma. Graphical user interfaces for R. *Journal of Statistical Software*, 49(1): 1–8, 2012. URL http://www.jstatsoft.org/v49/i01/. [p189]

A. Zeileis and G. Grothendieck. zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27, 2005. URL http://www.jstatsoft.org/v14/i06/. [p192]

Milan Bouchet-Valat
Quantitative Sociology Laboratory (LSQ-CREST),
Centre for Studies in Social Change (OSC-Sciences Po & CNRS),
National Institute for Demographic Studies (INED)
France
nalimilan@club.fr

Gilles Bastin
Institute for Political Studies, University of Grenoble
Pacte Research Centre (CNRS)
France
gilles.bastin@sciencespo-grenoble.fr

Possible Directions for Improving Dependency Versioning in R

by Jeroen Ooms

Abstract One of the most powerful features of R is its infrastructure for contributed code. The built-in package manager and complementary repositories provide a great system for development and exchange of code, and have played an important role in the growth of the platform towards the de-facto standard in statistical computing that it is today. However, the number of packages on CRAN and other repositories has increased beyond what might have been foreseen, and is revealing some limitations of the current design. One such problem is the general lack of dependency versioning in the infrastructure. This paper explores this problem in greater detail, and suggests approaches taken by other open source communities that might work for R as well. Three use cases are defined that exemplify the issue, and illustrate how improving this aspect of package management could increase reliability while supporting further growth of the R community.

Package management in R

One of the most powerful features of R is its infrastructure for contributed code (Fox, 2009). The base R software suite that is released several times per year ships with the *base* and *recommended* packages and provides a solid foundation for statistical computing. However, most R users will quickly resort to the package manager and install packages contributed by other users. By default, these packages are installed from the "Comprehensive R Archive Network" (**CRAN**), featuring over 4300 contributed packages as of 2013. In addition, other repositories like BioConductor (Gentleman et al., 2004) and Github (Dabbish et al., 2012) are hosting a respectable number of packages as well.

The *R Core team* has done a tremendous job in coordinating the development of the base software along with providing, supporting, and maintaining an infrastructure for contributed code. The system for sharing and installing contributed packages is easily taken for granted, but could in fact not survive without the commitment and daily efforts from the repository maintainers. The process from submission to publication of a package involves several manual steps needed to ensure that all published packages meet standards and work as expected, on a variety of platforms, architectures and R versions. In spite of rapid growth and limited resources, CRAN has managed to maintain high standards on the quality of packages. Before continuing, we want to express appreciation for the countless hours invested by volunteers in organizing this unique forum for statistical software. They facilitate the innovation and collaboration in our field, and unite the community in creating software that is both of the highest quality and publicly available. We want to emphasize that suggestions made in this paper are in no way intended as criticism on the status quo. If anything, we hope that our ideas help address some challenges to support further growth without having to compromise on the open and dynamic nature of the infrastructure.

The dependency network

Most R packages depend on one or more other packages, resulting in a complex network of recursive dependencies. Each package includes a 'DESCRIPTION' file which allows for declaration of several types of dependencies, including `Depends`, `Imports`, `Suggests` and `Enhances`. Based on the type of dependency relationship, other packages are automatically installed, loaded and/or attached with the requested package. Package management is also related to the issue of *namespacing*, because different packages can use identical names for objects. The 'NAMESPACE' file allows the developer to explicitly define objects to be exported or imported from other packages. This prevents the need to attach all dependencies and lookup variables at runtime, and thereby decreases chances of masking and naming-conflicts. Unfortunately, many packages are not taking advantage of this feature, and thereby force R to attach all dependencies, unnecessarily filling the search path of a session with packages that the user has not asked for. However, this is not the primary focus of this paper.

Package versioning

Even though CRAN consistently archives older versions of every package when updates are published, the R software itself takes limited advantage of this archive. The package manager identifies packages by name only when installing or loading a package. The `install.packages` function downloads and installs the *current* version of a CRAN package into a single global library. This library contains a

single version of each package. If a previous version of the package is already installed on the system, it is overwritten without warning. Similarly, the `library` function will load the earliest found package with a matching name.

The 'DESCRIPTION' file does allow the package author to specify a certain version of a dependency by postfixing the package name with >=, <= or == and a version string. However, using this feature is actually dangerous because R might not be able to satisfy these conditions, causing errors. This is again the result of R libraries, sessions and repositories being limited to a single current version of each package. When a package would require a version of a dependency that is not already installed or current on CRAN, it can not be resolved automatically. Furthermore, upgrading a package in the global library to the current CRAN version might break other packages that require the previously installed version. Experienced R users might try to avoid such problems by manually maintaining separate libraries for different tasks and projects. However, R can still not have multiple versions of a package loaded concurrently. This is perhaps the most fundamental problem because it is nearly impossible to work around. If package authors would actually declare specific versions of dependencies, any two packages requiring different versions of one and the same dependency will conflict and cannot be used together. In practice, this limitation discourages package authors to be explicit about dependency versions. The >= operator is used by some packages, but it only checks if an installed dependency is outdated and needs to be synchronized with CRAN. It still assumes that any current or future version will suffice, and does not protect packages from breaking when their dependency packages change. The <= and == operators are barely used at all.

When identifying a package by its name only, we implicitly make the assumption that different versions of the package are interchangeable. This basic assumption has far-reaching implications and consequences on the distributed development process and reliability of the software as a whole. In the context of the increasingly large pool of inter-dependent packages, violations of this assumption are becoming increasingly apparent and problematic. In this paper we explore this problem is greater detail, and try to make a case for moving away from this assumption, towards systematic versioning of dependency relationships. The term *dependency* in this context does not exclusively refer to formally defined relations between R packages. Our interpretation is a bit more general in the sense that any R script, Sweave document, or third party application *depends* on R and certain packages that are needed to make it function. The paper is largely motivated by personal experiences, as we have come to believe that limitations of the current dependency system are underlying multiple problems that R users and developers might experience. Properly addressing these concerns could resolve several lingering issues at once, and make R a more reliable and widely applicable analytical engine.

Use cases

A dependency defines a relationship wherein a certain piece of software requires some other software to run or compile. However, software constantly evolves, and in the open source world this happens largely unmanaged. Consequently, any software library might actually be something different today than it was yesterday. Hence, solely defining the dependency relationship in terms of the name of the software is often insufficient. We need to be more specific, and declare explicitly which version(s), branch(es) or release(s) of the other software package will make our program work. This is what we will refer to as *dependency versioning*.

This problem is not at all unique to R; in fact a large share of this paper consists of taking a closer look at how other open source communities are managing this process, and if some of their solutions could apply to R as well. But first we will elaborate a bit further on how this problem exactly appears in the context of R. This section describes three use cases that reveal some limitations of the current system. These use cases delineate the problem and lead towards suggestions for improvements in subsequent sections.

Case 1: Archive / repository maintenance

A medium to large sized repository with thousands of packages has a complicated network of dependencies between packages. CRAN is designed to consider the very latest version of every package as the only *current* version. This design relies on the assumption that at any given time, the latest versions of all packages are compatible. Therefore, R's built-in package manager can simply download and install the current versions of all dependencies along with the requested package, which seems convenient. However, to developers this means that every package *update* needs to maintain full backward compatibility with all previous versions. No version can introduce any breaking changes, because other packages in the repository might be relying on things in a certain way. Functions or objects may never be removed or modified; names, arguments, behavior, etc, must remain the same. As the dependency network gets larger and more complex, this policy becomes

increasingly vulnerable. It puts a heavy burden on contributing developers, especially the popular ones, and results in increasingly large packages that are never allowed to deprecate or clean up old code and functionality.

In practice, the assumption is easily violated. Every time a package update is pushed to CRAN, there is a real chance of some reverse dependencies failing due to a breaking change. In the case of the most popular packages, the probability of this happening is often closer to 1 than to 0, regardless of the author. Uwe Ligges has stated in his keynote presentation at useR that CRAN automatically detects some of these problems by rebuilding every package up in the dependency tree. However, only a small fraction of potential problems reveal themselves during the build of a package, and when found, there is no obvious solution. One recent example was the forced roll-back of the **ggplot2** (Wickham, 2009) update to version 0.9.0, because the introduced changes caused several other packages to break. The author of the **ggplot2** package has since been required to announce upcoming updates to authors of packages that depend on **ggplot2**, and provide a release candidate to test compatibility. The dependent packages are then required to synchronize their releases if any problems arise. However, such manual solutions are far from flawless and put even more work on the shoulders of contributing developers. It is doubtful that all package authors on CRAN have time and resources to engage in an extensive dialogue with other maintainers for each update of a package. We feel strongly that a more systematic solution is needed to guarantee that software published on CRAN keeps working over time; current as well as older versions.

When the repository reaches a critical size, and some packages collect hundreds of reverse dependencies, we have little choice but to acknowledge the fact that every package has only been developed for, and tested with, certain versions of its dependencies. A policy of assuming that any current or future version of a dependency should suffice is dangerous and sets the wrong incentives for package authors. It discourages change, refactoring or cleanup, and results in packages accumulating an increasingly heavy body of legacy code. And as the repository grows, it is inevitable that packages will nevertheless eventually break as part of the process. What is needed is a redesign that supports the continuous decentralized change of software and helps facilitate more reliable package development. This is not impossible: there are numerous open source communities managing repositories with more complex dependency structures than CRAN. Although specifics vary, they form interesting role models to our community. As we will see later on, a properly archived repository can actually come to be a great asset rather than a liability to the developer.

Case 2: Reproducibility

Replication is the ultimate standard by which scientific claims are judged. However, complexity of data and methods can make this difficult to achieve in computational science (Peng, 2011). As a leader in scientific computing, R takes a pioneering role in providing a system that encourages researchers to strive towards the gold standard. The CRAN Task View on Reproducible Research states that:

> *The goal of reproducible research is to tie specific instructions to data analysis and experimental data so that scholarship can be recreated, better understood and verified.*

In R, reproducible research is largely facilitated using literate programming techniques implemented in packages like **Sweave** that mix (weave) R code with LaTeX-markup to create a "reproducible document" (Leisch, 2002). However, those ever faced with the task of actually reproducing such a document might have experienced that the Sweave file does not always compile out of the box. Especially if it was written several years ago and loads some contributed packages, chances are that essential things have changed in the software since the document was created. When we find ourselves in such a situation, recovering the packages needed to reproduce the document might turn out to be non-trivial.

An example: suppose we would like to reproduce a Sweave document which was created with R 2.13 and loads the **caret** package (Kuhn, 2013). If no further instructions are provided, this means that any of the approximately 25 releases of **caret** in the life cycle of R 2.13 (April 2011 to February 2012) could have been used, making reproducibility unlikely. Sometimes authors add comments in the code where the package is loaded, stating that e.g. **caret 4.78** was used. However, this information might also turn out to be insufficient: **caret** depends on 4 packages, and suggests another 59 packages, almost all of which have had numerous releases in R 2.13 time frame. Consequently, **caret 4.78** might not work anymore because of changes in these dependencies. We then need to do further investigation to figure out which versions of the dependency packages were current at the time of the **caret 4.78** release. Instead, lets assume that the prescient researcher anticipated all of this, and saved the full output of `sessionInfo()` along with the Sweave document, directly after it was compiled. This output lists the version of each loaded package in the active R session. We could then proceed by manually downloading and installing R 2.13 along with all of the required packages from the archive. However, users on a commercial operating systems might be up for another surprise: unlike source packages,

binary packages are not fully archived. For example, the only binary builds available for R 2.13 are respectively **caret 5.13** on Windows, and **caret 5.14** on OSX. Most likely, they will face the task of rebuilding each of the required packages from source in an attempt to reconstruct the environment of the author.

Needless to say, this situation is suboptimal. For manually compiling a single Sweave document we might be willing to make this effort, but it does not provide a solid foundation for systematic or automated reproducible software practices. To make results generated by R more reproducible, we need better conventions and/or native support that is both explicit and specific about contributed code. For an R script or Sweave document to stand the test of time, it should work at least on the same version of R that was used by the author. In this respect, R has higher requirements on versioning than other software. Reproducible research does not just require a version that will make things work, but one that generates exactly the same output. In order to systematically reproduce results R, package versions either need to be standardized, or become a natural part of the language. We realize this will not archive perfect reproducibility, as problems can still arise due to OS or compiler specific behavior. However, it will be a major step forward that has the potential of turning reproducibility into a natural feature of the software, rather than a tedious exercise.

Case 3: Production applications

R is no longer exclusively used by the local statistician through an interactive console. It is increasingly powering systems, stacks and applications with embedded analytics and graphics. When R is part of say, an application used in hospitals to create on-demand graphics from patient data, the underlying code needs to be stable, reliable, and redistributable. Within such an application, even a minor change in code or behavior can result in complete failure of the system and cannot easily be fixed or debugged. Therefore, when an application is put in production, software has to be completely frozen.

An application that builds on R has been developed and tested with certain versions of the base software and R packages used by the application. In order to put this application in production, exactly these versions need to be shipped, installed and loaded by the application on production servers. Managing, distributing and deploying production software with R is remarkably hard, due to limited native dependency versioning and the single global library design. Administrators might discover that an application that was working in one place does not work elsewhere, even though exactly the same operating system, version of R, and installation scripts were used. The problem of course is that the contributed packages constantly change. Problems become more complicated when a machine is hosting many applications that were developed by different people and depend on various packages and package versions.

The default behavior of loading packages from a global library with bleeding edge versions is unsuitable for building applications. Because the CRAN repository has no notion of stable branches, one manually needs to download and install the correct versions of packages in a separate library for each application to avoid conflicts. This is quite tricky and hard to scale when hosting many applications. In practice, application developers might not even be aware of these pitfalls, and design their applications to rely on the default behavior of the package manager. They then find out the hard way that applications start breaking down later on, because of upstream changes or library conflicts with other applications.

Solution 1: staged distributions

The problem of managing bottom-up decentralized software development is not new; rather it is a typical feature of the open source development process. The remainder of this paper will explore two solutions from other open source communities, and suggest how these might apply to R. The current section describes the more classic solution that relies on staged software *distributions*.

A *software distribution* (also referred to as a *distribution* or a *distro*) is a collection of software components built, assembled and configured so that it can be used essentially "as is" for its intended purpose. Maintainers of distributions do not develop software themselves; they collect software from various sources, package it up and redistribute it as a system. Distributions introduce a formal release cycle on the continuously changing upstream developments and maintainers of a distribution take responsibility for ensuring compatibility of different packages within a certain release of the distribution. Software distributions are most commonly known in the context of free operating systems (BSD, Linux, etc). Staging and shipping software in a distribution has proven to scale well to very large code bases. For example, the popular Debian GNU/Linux distribution (after which R's package description format was modeled) features over 29000 packages with a large and complex dependency network. No single person is familiar with even a fraction of the code base that is hosted in this

repository. Yet through well organized staging and testing, this distribution is known to be one of the most reliable operating systems today, and is the foundation for a large share of the global IT infrastructure.

The release cycle

In a nutshell, a staged distribution release can be organized as follows. At any time, package authors can upload new versions of packages to the *devel* pool, also known as the *unstable* branch. A release cycle starts with distribution maintainers announcing a *code freeze* date, several months in advance. At this point, package authors are notified to ensure that their packages in the unstable branch are up to date, fix bugs and resolve other problems. At the date of the code freeze, a copy (fork) of the unstable repository is made, named and versioned, which goes into the *testing* phase. Software in this branch will then be subject to several iterations of intensive testing and bug fixing, sometimes accompanied by *alpha* or *beta* releases of the distribution. However, software versions in the testing branch will no longer receive any major updates that could potentially have side effects or break other packages. The goal is to converge to increasingly stable set of software. When after several testing rounds the distribution maintainers are confident that all serious problems are fixed, the branch is tagged *stable* and released to the public. Software in a stable release will usually only receive minor non-breaking updates, like important compatibility fixes and security updates. For the next "major release" of any software, the user will have to wait for the next cycle of the distribution. As such, everyone using a certain release of the distribution is using exactly the same versions of all programs and libraries on the system. This is convenient for both users and developers and gives distributions a key role in bringing decentralized open source development efforts together.

R: downstream staging and repackaging

The semi annual releases of the r-base software suite can already be considered as a distribution of the 29 `base` and `recommended` packages. However in the case of R, this collection is limited to software that has been centrally developed and released by the same group of people; it does not include contributed code. Due to the lack of native support for dependency versioning in R, several third party projects have introduced some form of downstream staging in order to create stable, redistributable collections R software. This section lists some examples and explains why this is suboptimal. In the next section we will discuss what would be involved with extending the R release cycle to contributed packages.

One way of staging R packages downstream is by including them in existing software distributions. For example, Eddelbuettel and Blundell (2009) have wrapped some popular CRAN packages into `deb` packages for the Debian and Ubuntu systems. Thereby, pre-compiled binaries are shipped in the distribution along with the R base software, putting version compatibility in the hands of the maintainers (among other benefits). This works well, but requires a lot of effort and commitment from the package maintainer, which is why this has only been done for a small subset of the CRAN packages. Most distributions expect high standards on the quality of the software and package maintenance, which makes this approach hard to scale up to many more packages. Furthermore, we are tied to the release cycle of the distribution, resulting in a somewhat arbitrary and perhaps unfortunate snapshot of CRAN packages when the distribution freezes. Also, different distributions will have different policies on if, when and which packages they wish to ship with their system.

Another approach is illustrated by domain-specific projects like BioConductor (genomic data) and REvolution R Enterprise (big data). Both these systems combine a fixed version of R with a custom library of frozen R packages. In the case of REvolution, the full library is included with the installer; for BioConductor they are provided through a dedicated repository. In both cases, this effectively prevents installed software from being altered unexpectedly by upstream changes. However, this also leads to a split in the community between users of R, BioConductor, and REvolution Enterprise. Because of the differences in libraries, R code is not automatically portable between these systems, leading to fragmentation and duplication of efforts. E.g. BioConductor seems to host many packages that could be more generally useful; yet they are unknown to most users of R. Furthermore, both projects only target a limited set of packages; they still rely on CRAN for the majority of the contributed code.

The goal of staging is to tie a fixed set of contributed packages to a certain release of R. If these decisions are passed down to distributions or organizations, a multitude of local conventions and repositories arise, and different groups of users will still be using different package versions. This leads to unnecessary fragmentation of the community by system, organization, or distribution channel. Moreover, it is often hard to assess compatibility of third party packages, resulting in somewhat arbitrary local decision making. It seems that the people who are in the best position to manage and control compatibility are the package authors themselves. This leads us to conclude that a more

appropriate place to organize staging of R packages is further upstream.

Branching and staging in CRAN itself

Given that the community of R contributors evolves mainly around CRAN, the most desirable approach to organizing staging would be by integrating it with the publication process. Currently, CRAN is managed as what distributions would consider a *development* or *unstable* branch. It consists of the pool of *bleeding-edge* versions, straight from package authors. Consequently it is wise to assume that software in this branch might break on a regular basis. Usually, the main purpose of an *unstable* branch is for developers to exchange new versions and test compatibility of software. Regular users obtain software releases from *stable* branches instead. This does not sound unfamiliar: the r-base software also distinguishes between stable versions *r-release* and *r-release-old*, and an unstable development version, *r-devel*.

The fact that R already has an semi-annual release cycle for the 29 `base` and `recommended` packages, would make it relatively straightforward to extend this cycle to CRAN packages. A snapshot of CRAN could be frozen along with every version of *r-release*, and new package updates would only be published to the *r-devel* branch. In practice, this could perhaps quite easily be implemented by creating a directory on CRAN for each release of R, containing symbolic links to the versions of the packages considered *stable* for this release. In the case of binary packages for OSX and Windows, CRAN actually already has separate directories with builds for each release of R. However currently these are not frozen and continuously updated. In a staged repository, newly submitted packages are only build for the current *devel* and *testing* branches; they should not affect *stable* releases. Exceptions to this process could still be granted to authors that need to push an important update or bugfix within a stable branch, commonly referred to as *backporting*, but this should only happen incidentally.

To fully make the transition to a staged CRAN, the default behavior of the package manager must be modified to download packages from the stable branch of the current version of R, rather than the latest development release. As such, all users on a given version of R will be using the same version of each CRAN package, regardless on when it was installed. The user could still be given an option to try and install the development version from the unstable branch, for example by adding an additional parameter to `install.packages` named `devel=TRUE`. However when installing an unstable package, it must be flagged, and the user must be warned that this version is not properly tested and might not be working as expected. Furthermore, when loading this package a warning could be shown with the version number so that it is also obvious from the output that results were produced using a non-standard version of the contributed package. Finally, users that would always like to use the very latest versions of all packages, e.g. developers, could install the `r-devel` release of R. This version contains the latest commits by R Core and downloads packages from the devel branch on CRAN, but should not be used or in production or reproducible research settings.

Organizational change

Appropriate default behavior of the software is a key element to encourage adoption of conventions and standards in the community. But just as important is communication and coordination between repository maintainers and package authors. To make staging work, package authors must be notified of upcoming deadlines, code freezes or currently broken packages. Everyone must realize that the package version that is current at the time of code freeze, will be used by the majority of users of the upcoming version of R. Updates to already released *stable* branches can only be granted in exceptional circumstances, and must guarantee to maintain full backward compatibility. The policies of the BioConductor project provide a good starting point and could be adapted to work for CRAN.

Transitioning to a system of "stable" and "development" branches in CRAN, where the stable branch is conventional for regular users, could tremendously improve the reliability of the software. The version of the R software itself would automatically imply certain versions of contributed packages. Hence, all that is required to reproduce a Sweave document created several years ago, is which version of R was used to create the document. When deploying an application that depends on R 2.15.2 and various contributed packages, we can be sure that a year later the application can be deployed just as easily, even though the authors of contributed packages used by the application might have decided to implement some breaking changes. And package updates that deprecate old functionality or might break other packages that depend on it, can be uploaded to the *unstable* branch without worries, as the stable branches will remain unchanged and users won't be affected. The authors of the dependent packages that broke due to the update can be warned and will have sufficient time to fix problems before the next *stable* release.

Solution 2: versioned package management

The previous section described the "classical" solution of creating distributable sets of compatible, stable software. This is a proven approach and has been adopted in some way or another by many open-source communities. However, one drawback of this approach might be that some additional coordination is needed for every release. Another drawback is that it makes the software a bit more conservative, in the sense that regular users will generally be using versions of packages that are at least a couple of months old. The current section describes a different approach to the problem that is used by for example the Javascript community. This method is both reliable and flexible, however would require some more fundamental changes to be implemented in R.

Node.js and NPM

One of the most recent and fastest growing open source communities is that of the node.js software (for short: *node*), a Javascript server system based on the open source engine *V8* from Google. One of the reasons that the community has been able to grow rapidly is because of the excellent package manager and identically named repository, *NPM*. Even though this package manager is only 3 years old, it is currently hosting over 30000 packages with more than a million downloads daily, and has quickly become the standard way of distributing Javascript code. The NPM package manager is a powerful tool for development, publication and deployment of both libraries and applications. NPM addresses some problems that Javascript and R actually have in common, and makes an interesting role model for a modern solution to the problem.

The Javascript community can be described as decentralized, unorganized and highly fragmented development without any quality control authority. Similar to CRAN, NPM basically allows anyone to claim a "package name" and start publishing packages and updates to the repositories. The repository has no notion of branches and simply stores every version of a package indefinitely in its archives. However, a major difference with R is how the package manager handles installation, loading and namespacing of packages.

Dependencies in NPM

Every *NPM* package ships with a file named 'package.json', which is the equivalent of the 'DESCRIPTION' in R packages, yet a bit more advanced. An overview of the full feature set of the package manager is beyond the scope of this paper, but the interested reader is highly encouraged to take a look over the fence at this well designed system: https://npmjs.org/doc/json.html. The most relevant feature in the context CRAN is how NPM declares and resolves dependencies.

Package dependencies are defined using a combination of the package *name* and *version range descriptor*. This descriptor is specified with a simple dedicated syntax, that extends some of the standard versioning notation. Below a snippet taken from the 'package.json' file in the NPM manual:

```
"dependencies" : {
   "foo" : "1.0.0 - 2.9999.9999",
   "bar" : ">=1.0.2 <2.1.2",
   "baz" : ">1.0.2 <=2.3.4",
   "boo" : "2.0.1",
   "qux" : "<1.0.0 || >=2.3.1 <2.4.5",
   "asd" : "http://asdf.com/asdf.tar.gz",
   "til" : "~1.2",
   "elf" : "~1.2.3",
   "two" : "2.x",
   "thr" : "3.3.x",
}
```

The version range descriptor syntax is a powerful tool to specify which version(s) or version range(s) of dependencies are required. It provides the exact information needed to build, install and/or load

the software. In contrast to R, NPM takes full advantage of this information. In R, all packages are installed in one or more global libraries, and at any given time a subset of these packages is loaded in memory. This is where NPM takes a very different approach. During installation of a package, NPM creates a *subdirectory* for dependencies inside the installation directory of the package. It compares the list of dependency declarations from the 'package.json' with an index of the repository archive, and then constructs a private library containing the full dependency tree and precise versions as specified by the author. Hence, every installed package has its own library of dependencies. This works recursively, i.e. every dependency package inside the library again has its own dependency library.

```
jeroen@ubuntu:~/Desktop$ npm install d3
jeroen@ubuntu:~/Desktop$ npm list
/home/jeroen/Desktop
└─┬ d3@2.10.3
  ├─┬ jsdom@0.2.14
  │ ├─┬ contextify@0.1.3
  │ │ └── bindings@1.0.0
  │ ├── cssom@0.2.5
  │ ├── htmlparser@1.7.6
  │ └─┬ request@2.12.0
  │   ├─┬ form-data@0.0.3
  │   │ ├── async@0.1.9
  │   │ └─┬ combined-stream@0.0.3
  │   │   └── delayed-stream@0.0.5
  │   └── mime@1.2.7
  └── sizzle@1.1.0
```

By default, a package loads dependencies from its private library, and the namespace of the dependency is imported explicitly in the code. This way, an installed NPM package is completely unaffected by other applications, packages, and package updates being installed on the machine. The private library of any package contains all required dependencies, with the exact versions that were used to develop the package. A package or application that has been tested to work with certain versions of its dependencies, can easily be installed years later on another machine, even though the latest versions of dependencies have had major changes in the mean time.

Back to R

A similar way of managing packages could be very beneficial to R as well. It would enable the same dynamic development and stable installation of packages that has resulted in a small revolution within the Javascript community. The only serious drawback of this design is that it requires more disk space and slightly more memory, due to multiple versions of packages being installed and/or loaded. Yet the memory required to load an additional package is minor in comparison with loading and manipulating a medium sized dataset. Considering the wide availability of low cost disk space and memory these days, we expect that most users and developers will happily pay this small price for more reliable software and reduced debugging time.

Unfortunately, implementing a package manager like NPM for R would require some fundamental changes in the way R installs and loads packages and namespaces, which might break backward compatibility at this point. One change that would probably be required for this is to move away from the `Depends` relation definition, and require all packages to rely on `Imports` and a `NAMESPACE` file to explicitly import objects from other packages. A more challenging problem might be that R should be able to load multiple versions of a package simultaneously while keeping their namespaces separated. This is necessary for example when two packages are in use, which both depend on different versions of one and the same third package. In this case, the objects, methods and classes exported by the dependency package should affect only the package that imported them.

Finally, it would be great if the package manager was capable of installing multiple versions of a package inside a library, for example by appending the package version to the name of the installation directory (e.g. `MASS_7.3-22`). The `library` and `require` functions could then be extended with an argument specifying the version to be loaded. This argument could use the same version range descriptor syntax that packages use to declare dependencies. Missing versions could automatically be installed, as nothing gets overwritten.

```
library(ggplot2, version="0.8.9")
library(MASS, version="7.3-x")
library(Matrix, version=">=1.0")
```

Code as above leaves little ambiguity and tremendously increases reliability and reproducibility of R code. When the code is explicit about which package versions are loaded, and packages are explicit about dependency versions, an R script or Sweave document that once worked on a certain version of R, will work for other users, on different systems, and keep working over time, regardless of upstream changes. For users not concerned with dependency versioning, the default value of the `version` argument could be set to `"*"`. This value indicates that any version will do, in which case the package manager gives preference to the most recent available version of the package.

The benefits of a package manager capable of importing specific versions of packages would not just be limited to contributed code. Such a package manager would also reduce the necessity to include all of the standard library and more in the R releases. If implemented, the R Core team could consider moving some of the *base* and *recommended* packages out of the `r-base` distribution, and offer them exclusively through CRAN. This way, the R software could eventually become the minimal core containing only the language interpreter and package manager, similar to e.g. Node and NPM. More high-level functionality could be loaded on demand as versioning is controlled by the package manager. This would allow for less frequent releases of the R software itself, and further improve compatibility and reproducibility between versions of R.

Summary

The infrastructure for contributed code has supported the steady growth and adoption of the R software. For the majority of users, contributed code is just as essential in their daily work as the R base software suite. But the number of packages on CRAN has grown beyond what could have been foreseen, and practices and policies that used to work on a smaller scale are becoming unsustainable. At the same time there is an increasing demand for more reliable, stable software, that can be used as part of embedded systems, enterprise applications, or reproducible research. The design and policies of CRAN and the package manager shape the development process and play an important role in determining the future of the platform. The current practice of publishing package updates directly to end-users facilitates a highly versatile development, but comes at the cost of reliability. The default behavior of R to install packages in a single library with only the latest versions is perhaps more appropriate for developers than regular users. After nearly two decades of development, R has reached a maturity where a slightly more conservative approach could be beneficial.

This paper explained the problem of dependency versioning, and tried to make a case for transitioning to a system that does not assume that package versions are interchangeable. The most straightforward approach would be by extending the *r-release* and *r-devel* branches to the full CRAN repository, and only publish updates of contributed packages to the *r-devel* branch of R. This way, the *stable* versions of R are tied to a fixed version of each CRAN package, making the code base and behavior of a given release of R less ambiguous. Furthermore, a release cycle allows us to concentrate coordination and testing efforts for contributed packages along with releases of R, rather than continuously throughout the year.

In the long term, a more fundamental revision of the packaging system could be considered, in order to facilitate dynamic contributed development without sacrificing reliability. However, this would involve major changes in the way libraries and namespaces are managed. The most challenging problem will be support for concurrently loading multiple versions of a package. But when the time is ready to make the jump to the next major release of R, we hope that R Core will consider revising this important part of the software, adopting modern approaches and best practices of package management that are powering collaboration and uniting efforts within other open source communities.

Bibliography

L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012. [p197]

D. Eddelbuettel and C. Blundell. cran2deb: A fully automated CRAN to Debian package generation system. Presented at UseR Conference, July 10-12, Rennes, 2009. URL https://r-forge.r-project.org/projects/cran2deb/. [p201]

J. Fox. Aspects of the Social Organization and Trajectory of the R project. *The R Journal*, 1(2):5–13, 2009. [p197]

R. Gentleman, V. Carey, D. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5(10):R80, 2004. [p197]

M. Kuhn. *caret: Classification and Regression Training*, 2013. URL http://CRAN.R-project.org/package=caret. R package version 5.16-04. [p199]

F. Leisch. Sweave. Dynamic generation of statistical reports using literate data analysis. *Report Series SFB "Adaptive Information Systems and Modelling in Economics and Management Science"*, 2002. URL http://epub.wu.ac.at/1788/1/document.pdf. [p199]

R. D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011. [p199]

H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book. [p118, 199]

Jeroen Ooms
Department of Statistics
University of California
Los Angeles
jeroen.ooms@stat.ucla.edu
http://jeroenooms.github.io/

Translating Probability Density Functions: From R to BUGS and Back Again

by David S. LeBauer, Michael C. Dietze, Benjamin M. Bolker

Abstract The ability to implement statistical models in the BUGS language facilitates Bayesian inference by automating MCMC algorithms. Software packages that interpret the BUGS language include OpenBUGS, WinBUGS, and JAGS. R packages that link BUGS software to the R environment, including **rjags** and **R2WinBUGS**, are widely used in Bayesian analysis. Indeed, many packages in the Bayesian task view on CRAN (`http://cran.r-project.org/web/views/Bayesian.html`) depend on this integration. However, the R and BUGS languages use different representations of common probability density functions, creating a potential for errors to occur in the implementation or interpretation of analyses that use both languages. Here we review different parameterizations used by the R and BUGS languages, describe how to translate between the languages, and provide an R function, `r2bugs.distributions`, that transforms parameterizations from R to BUGS and back again.

Distribution	Lang.	Parameterization	Use	Notes
Normal				
	R	$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	`dnorm(`x`, mean =` μ`, sd =` σ`)`	
	BUGS	$\sqrt{\frac{\tau}{2\pi}}\exp\left(-(x-\mu)^2\tau\right)$	`dnorm(mean =` μ`, precision =` τ`)`	$\tau=\left(\frac{1}{\sigma}\right)^2$
log-Normal				
	R	$\frac{1}{\sqrt{2\pi}\sigma x}\exp\left(-\frac{(\log(x)-\mu)^2}{(2\sigma^2)}\right)$	`dlnorm(`x`, mean =` μ`, sd =` σ`)`	
	BUGS	$\frac{\sqrt{\tau}}{x}\exp\left(\frac{-\tau(\log(x)-\mu)^2}{2}\right)$	`dlnorm(mean =` μ`, precision =` τ`)`	$\tau=\left(\frac{1}{\sigma}\right)^2$
Binomial				reverse parameter order
	R	$\binom{n}{x}p^x(1-p)^{n-x}$	`dbinom(`x`, size =` n`, prob =` p`)`	
	BUGS	same	`dbin(prob =` p`, size =` n`)`	
Negative Binomial				reverse parameter order
	R	$\frac{\Gamma(x+n)}{\Gamma(n)x!}p^n(1-p)^x$	`dnbinom(`x`, size =` n`, prob =` p`)`	size (`n`) is continuous
	BUGS	$\binom{x+r-1}{x}p^r(1-p)^x$	`dnegbin(prob =` p`, size =` r`)`	size (`r`) is discrete
Weibull				
	R	$\frac{a}{b}\left(\frac{x}{b}\right)^{a-1}\exp\left(-\left(\frac{x}{b}\right)^a\right)$	`dweibull(`x`, shape =` a`, scale =` b`)`	
	BUGS	$\nu\lambda x^{\nu-1}\exp\left(-\lambda x^\nu\right)$	`dweib(shape =` ν`, lambda =` λ`)`	$\lambda=\left(\frac{1}{b}\right)^a$
Gamma				reverse parameter order
	R	$\frac{r^a}{\Gamma(a)}x^{a-1}\exp(-xr)$	`dgamma(`x`, shape = a, rate =` r`)`	
	BUGS	$\frac{\lambda^r x^{r-1}\exp(-\lambda x)}{\Gamma(r)}$	`dgamma(shape =` r`, lambda =` λ`)`	

Table 1: Summary of different parameterizations of common distributions used by R and BUGS. **Note:** For ease of reference, parameterizations follow the JAGS and R documentation; as a result, the table includes equivalent equations that appear different, either because JAGS and R use different names for the same parameter or because the equation has been rearranged. For example, the shape parameter of the *Gamma* distribution is r in the BUGS documentation and a in the R documentation. For the *Binomial*, *Negative Binomial*, and *Gamma* distributions, BUGS and R expect parameters in different order (the order of parameters matters since arguments are assigned based on position in BUGS and may be in R as well). R allows alternate parameterizations for the *Negative Binomial* and *Gamma* distributions, but these are not shown here. The variable x is implicit in all of the BUGS "Use" expressions. The *Beta*, *Poisson*, *Exponential*, and *Uniform* distributions have identical parameterizations in R and BUGS.

Probability density functions in R and BUGS

R and BUGS implement many of the same probability distribution functions, but they often parameterize the same distribution differently (Table 1). Although these probability distribution functions are clearly described in the documentation of their respective languages, we were unable to find a summary of these differences in one place. The motivation for this article is to document and clarify these differences. Our sources are the JAGS documentation (Plummer, 2010) and the documentation of individual R functions.

A bilingual translation function

To support the automation of model specification in JAGS with priors computed and stored in R (LeBauer et al., 2013), we developed a function to translate parameterizations of common probability distributions from R to BUGS (and back again, by specifying `direction = 'bugs2r'`). Parameter transformations, parameter order, and differences in function names are documented in Table 1 and implemented in the R function `r2bugs.distributions`.

```
r2bugs.distributions <- function(priors, direction = 'r2bugs') {
  priors$distn  <- as.character(priors$distn)
  priors$parama <- as.numeric(priors$parama)
  priors$paramb <- as.numeric(priors$paramb)
  ## index dataframe according to distribution
  norm   <- priors$distn %in% c('norm', 'lnorm')    # these have same transform
  weib   <- grepl("weib", priors$distn)             # matches r and bugs version
  gamma  <- priors$distn == 'gamma'
  chsq   <- grepl("chisq", priors$distn)            # matches r and bugs version
  bin    <- priors$distn %in% c('binom', 'bin')     # matches r and bugs version
  nbin   <- priors$distn %in% c('nbinom', 'negbin') # matches r and bugs version

  ## Normal, log-Normal: Convert sd to precision
  exponent <- ifelse(direction == "r2bugs", -2, -0.5)
  priors$paramb[norm] <-  priors$paramb[norm] ^ exponent

  ## Weibull
  if(direction == 'r2bugs'){
    ## Convert R parameter b to BUGS parameter lambda by l = (1/b)^a
    priors$paramb[weib] <- (1 / priors$paramb[weib]) ^ priors$parama[weib]
  } else if (direction == 'bugs2r') {
    ## Convert BUGS parameter lambda to BUGS parameter b by b = l^(-1/a)
    priors$paramb[weib] <-  priors$paramb[weib] ^ (- 1 / priors$parama[weib] )
  }

  ## Reverse parameter order for binomial and negative binomial
  priors[bin | nbin, c('parama', 'paramb')] <-
    priors[bin | nbin, c('paramb', 'parama')]

  ## Translate distribution names
  if(direction == "r2bugs"){
    priors$distn[weib] <- "weib"
    priors$distn[chsq] <- "chisqr"
    priors$distn[bin]  <- "bin"
    priors$distn[nbin] <- "negbin"
  } else if(direction == "bugs2r"){
    priors$distn[weib] <- "weibull"
    priors$distn[chsq] <- "chisq"
    priors$distn[bin]  <- "binom"
    priors$distn[nbin] <- "nbinom"
  }
  return(priors)
}
```

A simple example

As an example, we take the R-parameterized prior distribution $X \sim \mathcal{N}(\mu = 10, \sigma = 2)$ and convert it to BUGS parameterization $X \sim \mathcal{N}(\mu = 10, \tau = 1/4)$. We specify a model in JAGS that allows us to sample directly from a prior distribution. The function works for each of the distributions in Table 1. This particular example is the JAGS implementation of '`rnorm(10000,10,2)`' in R. It is presented as minimal demonstration; for a non-trivial application, see LeBauer et al. (2013).

```
r.distn <- data.frame(distn = "norm", parama = 10, paramb = 2)
bugs.distn <- r2bugs.distributions(r.distn)

sample.bugs.distn <- function(prior = data.frame(distn = "norm", parama = 0,
                              paramb = 1), n = 10000) {
  require(rjags)
  model.string <- paste0(
    "model{Y ~ d", prior$distn,
    "(", prior$parama,
    ## chisqr has only one parameter
    ifelse(prior$distn == "chisqr", "", paste0(", ", prior$paramb)), ");",
    "a <- x}"
  )
  ## trick JAGS into running without data
  writeLines(model.string, con = "test.bug")
  j.model  <- jags.model(file = "test.bug", data = list(x = 1))
  mcmc.object <- window(
    coda.samples(
      model = j.model, variable.names = c('Y'),
      n.iter = n * 4, thin = 2),
    start = n)
  Y <- sample(as.matrix(mcmc.object)[,"Y"], n)
}
X <- sample.bugs.distn(bugs.distn)
```

Acknowlegements

This collaboration began on the Cross Validated statistical forum (http://stats.stackexchange.com/q/5543/1381). Funding was provided to DSL and MCD by the Energy Biosciences Institute.

Bibliography

D. S. LeBauer, D. Wang, K. T. Richter, C. C. Davidson, and M. C. Dietze. Facilitating feedbacks between field measurements and ecosystem models. *Ecological Monographs*, 83(2):133–154, 2013. URL http://www.esajournals.org/doi/abs/10.1890/12-0137.1. [p208, 209]

M. Plummer. JAGS Version 3.1.0 user manual, 2010. URL http://sourceforge.net/projects/mcmc-jags/. [p208]

David S. LeBauer
Energy Biosciences Institute
University of Illinois
USA
dlebauer@illinois.edu

Benjamin M. Bolker
Department of Mathematics & Statistics
McMaster University
Canada

Michael C. Dietze
Department of Earth And Environment
Boston University
USA

Conference Review: The 6th Chinese R Conference

by Jing Leng and Jingjing Guan

The 6th Chinese R Conference (Beijing session) was held in the Sinology Pavilion of Renmin University of China (RUC), Beijing, from May 18th to 19th, 2013. The conference was organized by the "Capital of Statistics" (COS, http://cos.name), an online statistical community in China. It was sponsored and co-organized by the Center for Applied Statistics of RUC, the School of Statistics of RUC, and the Business Intelligence Research Center of Peking University.

Since the 1st Chinese R Conference in 2008, this conference has become a regular and popular event for Chinese R users, where they share cutting edge techniques and applications with R. This is a bi-annual conference with a Beijing session in the summer and a Shanghai session in the winter each year.

This year, more than 400 attendees from China and overseas attended the Beijing conference. In particular, for the first time had two foreign speakers at the Beijing conference[1]: John Maindonald (Australian National University) and Graham Williams (Australian Taxation Office). Yihui Xie, the founder of COS and the Chinese R Conference, also came back from the United States and presented at the conference.

The conference program included nineteen talks on a variety of topics, including visualizations in epidemiology, customer behaviors of e-commerce websites, and text mining of online social networks. It also provided two lightning talk sessions as opportunities for people from different industries to promote their business and hire R users. Discussions among the audience and speakers showed increasing impact and popularity of R language in both the industry and academia in China. Below is the list of talks:

Opening Chen Yu, the Vice Chair of the conference, provided a brief introduction of the 6th Chinese R Conference; Prof Xizhi Wu, the first person to introduce R to the School of Statistics of RUC, gave an inspiring talk encouraging the younger generation to work hard (he mentioned Prof Bin Yu as an outstanding example); Prof Yanyun Zhao, the Dean of the School of Statistics of RUC, delivered a welcome speech;

Software Development "Sharing my lessons in R package development" by Yihui Xie; "A cloud-based decision making system based on R" by Ben-Chang Shia and Sizhe Liu;

Data Mining "Data mining with Rattle and R" by Graham Williams; "Detection of online public opinions: text mining and visualization in R" by He Wang;

Visualization "**displayHTS**: an R package for displaying data and results from high-throughput screening experiments" by Xiaohua Zhang; "An introduction to **MSToolkit**, **Rweibo** and **html5vis**: analysis of H7N9 in R" by Jian Li and Yang Zhou;

Business Applications "Application of R in eBay big data analysis" by Zhong Li and Jiaming Pan; "Data scientists and engineering applications of R" by Guozhu Wen; "Applications of machine learning in online advertisements" by Baotong Zhuang; "Quality assessment and intelligent sorting of user-generated content" by Hao Wang; "Online behavior in the mobile applications: an attempt in R" by Tingrui Zhou;

Statistical Methodologies "Bayesian hierarchical models in R and WinBUGS" by Xinhai Li; "Data cloning: easy maximum likelihood estimation for complex models: an application to zero-inflated responses of Internet ads" by Jingjing Guan; "On the ultrahigh dimensional linear discriminant analysis problem with a diverging number of classes" by Hansheng Wang;

[1] Thomas W. Yee, author of **VGAM**, was the first foreign speaker, and attended the Shanghai conference in 2011.

Kaleidoscope "Rethinking data analysis and data analysis tools" by John Maindonald; "Web scraping with R" by Nan Xiao; "An introduction to the Julia language" by Changyou Zhang.

In addition, a series of 5-minute talks for promotion purposes and R-related jobs were given by Merck China, 360buy, China Citic Bank, Careerfocus, Springer, SupStat, Alipay, Amazon, eBay, Baidu, and Douban, etc. Many companies have received a large number of job applications. After the lightning talks, the conference chair opened a clean R session, ran `sample(id,20)` and gave away 20 books in R and statistics (sponsored by publishers) to the lucky attendees.

Figure 1: Learning R without thinking it makes one confused; thinking R without learning it makes one shallow — Chenshun Lin, the chair of the lightning talk sessions, added R to a famous quote by Confucious and turned it to a hilarious pun.

The two-day event was a great success and we got many positive feedback messages from participants after the conference. We will further devote our future effort on:

- organizing more R meetings to meet the growing demand of R users in China;
- promoting statistics and data analysis in the industry;
- interacting with different fields, such as e-commerce, through more and advanced applications of R.

The conference summary and slides are freely available at http://cos.name/chinar/chinar-2013/ (in Chinese). We would like to thank the School of Statistics and the Center for Applied Statistics of RUC for their consistent support for the Chinese R Conference. We are also grateful to Prof Hansheng Wang for his encouragement and generous help. We appreciate the tremendous help of all student volunteers from RUC and COS. We look forward to the next R conference in China and warmly welcome more people to attend it. Inquiries and suggestions can be sent to chinar-committee@cos.name.

The conference committee consists of Tao Gao (Chair), Yu Chen (Vice Chair), Taiyun Wei, Sen Chen, Jianchong Su, Yanping Chen, Sizhe Liu, Yihui Xie, Manqi Xie, Zhanhang Xiao, Yishuo Deng, Yixuan Qiu, Yan Chen, Jing Leng.

Jing Leng
School of Statistics
Renmin University of China
Beijing, China P. R.
jing.leng@cos.name

Jingjing Guan
College of Business
City University of Hong Kong
Hong Kong SAR
jingjing.guan@cos.name

Conference Report: R/Finance 2013

by Joshua Ulrich

The fifth annual R/Finance conference for applied finance using R was held in Chicago, IL, USA on Friday May 17 and Saturday May 18, 2013.

The conference provided a venue to discuss how R can be used for portfolio management, time series analysis, advanced risk analysis, high-performance computing, market microstructure, and econometrics. As in prior years, the conference had 300 attendees from several countries (including several European countries, South Africa, Australia, and Russia). The program included seminars, keynotes, full-length talks, and lightning talks. The conference also provided exceptional networking opportunities.

Presentations

Five one-hour, single-track seminars were held on Friday morning:

- Whit Armstrong, Bryan Lewis: An Introduction to Distributed Computing in R
- Matthew Dowle: Advanced Tutorial on **data.table**
- Jan Humme, Brian Peterson: Using **quantstrat** to evaluate intraday trading strategies
- Dirk Eddelbuettel: Example-driven Introduction to **Rcpp**
- Jeffrey Ryan: R Programming for Financial Data

The first presentation was by keynote Ryan Sheftel, who talked about how he uses R on his bond trading desk. David Ardia showed how expected returns can be estimated via the covariance matrix. Ronald Hochreiter gave an overview of modeling optimization via his **modopt** package. Bernhard Pfaff used Bayesian utility optimization to allocate large portfolios.

Maria Belianina showed how R can interface with OneTick's high performance time series database. Yang Lu described Brinson-style portfolio attribution using **pa**. Michael Kapler used factor clusters to construct Risk Parity portfolios and evaluate risk contributions. Tammer Kamel gave a live demo of the **Quandl** package and said, "Quandl hopes to do to Bloomberg what Wikipedia did to Britannica."

Doug Martin talked about robust covariance estimation. Giles Heywood discussed several ways of estimating and forecasting covariance, and proposed an "open source equity risk and backtest system" as a means of matching talent with capital.

Ruey Tsay was the next keynote, and spoke about using principal volatility components to simplify multivariate volatility modeling. Alexios Ghalanos spoke about modeling multivariate time-varying skewness and kurtosis.

Kris Boudt examined changes in portfolio properties across volatility regimes. David Matteson described a new technique for detecting change points in any statistical property of univariate and multivariate time series. Celine Sun proposed a methodology to construct a full-rank covariance matrix using cross-sectional volatilities. Winston Chang gave a live demo of **shiny**.

Saturday started with Christian Silva, who evaluated statistical properties of moving-average-based strategies to determine when they do and don't work. He provided code at `http://rpubs.com/silvaac/6165`. Vyacheslav Arbuzov used a cluster of servers to analyze financial bubbles and crashes using an agent-based model. Stephen Rush examined the relationship between bond coupon and liquidity in different market regimes.

Samantha Azzarello discussed her work with Blu Putnam, which used a dynamic linear model to evaluate the Fed's performance vis-a-vis the Taylor Rule. Grant Cavanaugh examined the success of new ETF product listings. Jiahan Li used constrained least squares

on 4 economic fundamentals to forecast foreign exchange rates. Thomas Harte talked about regulatory requirements of foreign exchange pricing; basically documentation is important, Sweave to the rescue!

Sanjiv Das gave a keynote on 4 applications: 1) network analysis on SEC and FDIC filings to determine banks that pose systematic risk, 2) determining which home mortgage modification is optimal, 3) portfolio optimization with mental accounting, 4) venture capital communities.

Dirk Eddelbuettel showed how it's easy to write fast linear algebra code with **RcppArmadillo**. Klaus Spanderen showed how to use QuantLib from R, and even how to to call C++ from R from C++. Bryan Lewis talked about SciDB and the **scidb** package (SciDB contains fast linear algebra routines that operate on the database!).

Matthew Dowle gave an introduction to **data.table**. Chris Blakely showed a Java interface to R and a C implementation of the HEAVY realized volatility model. Mathieu Lestel described his 2012 Google Summer of Code project that added functionality to **PerformanceAnalytics**.

Attilio Meucci gave his keynote on visualizing advanced risk management and portfolio optimization. Immediately following, Brian Peterson gave a lightning on implementing Meucci's work in R (Attilio works in Matlab), which was part of a Google Summer of Code project last year.

Thomas Hanson presented his work with Don Chance (and others) on computational issues in estimating the volatility smile. Kam Hamidieh used options prices to recover the underlying asset's probability distribution estimate. Jeffrey Ryan showed how to manipulate options data in R with the **greeks** package.

Prizes

The conference wrapped up by giving away three books, generously donated by Springer, to three random people who submitted feedback surveys. The committee also presented the awards for best papers. The winners were:

- Regime switches in volatility and correlation of financial institutions, Boudt et. al.
- A Bayesian interpretation of the Federal Reserve's dual mandate and the Taylor Rule, Putnam & Azzarello
- Nonparametric Estimation of Stationarity and Change Points in Finance, Matteson et. al.
- Estimating High Dimensional Covariance Matrix Using a Factor Model, Sun (best student paper)

Networking

The two-hour conference reception at UIC on Friday was a great time to talk with speakers, and mingle with other attendees. Next was the (optional) dinner at The Terrace at Trump. Unfortunately, it was cold and windy, so we only spent 15-20 minutes on the terrace before moving inside. The food was fantastic, but the conversations were even better. After the final presentation on Saturday, many attendees continued conversations over food and drink at Jaks Tap.

Sponsors and organizers

The conference could not be successful without the support of our fantastic sponsors: International Center for Futures and Derivatives at UIC (our host), Revolution Analytics,

MS-Computational Finance at University of Washington, Google, lemnica, OpenGamma, OneMarketData, and RStudio.

Thanks to the committee: Gib Bassett, Peter Carl, Dirk Eddelbuettel, Brian Peterson, Dale Rosenthal, Jeffrey Ryan, Joshua Ulrich; and also to the event coordinators: Holly Griffin and Alexandrina Almazan.

Further information

The R/Finance website, http://www.RinFinance.com, contains information for past and future conferences. Slides (if made available by the authors) can be downloaded via the agenda page. We hope to see you in May 2014!

On Behalf of the Conference Committee,

Joshua Ulrich
FOSS Trading
Saint Louis, MO
USA
josh.m.ulrich@gmail.com

The R User Conference 2013

The ninth R user conference will take place at the University of Castilla-La Mancha, Albacete, Spain from Wednesday 10 July 2013 to Friday 12 July 2013. Following previous *useR!* conferences, this meeting of the R user community will

- focus on R as the 'lingua franca' of data analysis and statistical computing;
- provide a platform for R users to discuss and exchange ideas on how R can be used for statistical computation, data analysis, visualization and exciting applications in various fields;
- give an overview of the new features of the ever evolving R project.

As with the predecessor conferences, the program consists of two parts:

- invited talks discussing new R developments and exciting applications of R
- user-contributed presentation reflecting the wide range of fields in which R is used to analyze data.

A major goal of the useR! conference is to bring users from various fields together and provide a platform for discussion and exchange of ideas: both in the formal framework of presentations as well as in the informal times surrounding the conference sessions.

Invited speakers

The invited talks represent the spectrum of interest from important technical developments to exciting applications of R, presented by experts in the field:

- *María Jesús Bayarri*: New challenges and Bayes: The world of computer models.
- *José Manuel Benítez-Sánchez*: Computational Intelligence in R.
- *Duncan Murdoch*: What's new in R3.0.X.
- *Havard Rue*: Bayesian computing with INLA and the R-INLA package.
- *Steve Scott*: Bayesian computation in C++ with R as an interface.
- *Hadley Wickham*: BigR data

User-contributed sessions

In the contributed sessions, presenters will share innovative and interesting uses of R, covering topics such as:

- Bayesian statistics
- Bioinformatics
- Chemometrics and computational physics
- Data mining
- Econometrics & finance
- Environmetrics & ecological modeling
- High performance computing
- Imaging
- Interfaces with other languages/software
- Machine learning
- Multivariate statistics
- Nonparametric statistics
- Pharmaceutical statistics
- Psychometrics

- Spatial statistics
- Statistics in the social and political sciences
- Teaching
- Visualization & graphics

The poster session will be a major social event on the evening of the first day of the conference. Contributed talks will be organised in the following types of session:

- *useR! Kaleidoscope*: These sessions give a broad overview of the many different applications of R and should appeal to a wide audience.
- *useR! Focus Session*: These sessions cover topics of special interest and may be more technical.

In both cases presentations will be allowed 17 minutes, followed by 3 minutes discussion. In addition to the regular contributed talks, all participants are invited to present a *Lightning Talk*, for which no abstract is required. These talks provide a 5-minute platform to speak on any R-related topic and should particularly appeal to R newbies. Participants wishing to give such a talk must provide an informative title.

Pre-conference tutorials

Before the start of the official program, the following half-day tutorials will be offered on Tuesday, July 9th:

- *Esteban Alfaro, Matías Gámez and Noelia García*: Classification with Individual and Ensemble Trees
- *Jason Bryer and Robert Pruzek*: Introduction to Propensity Score Methods with R
- *Romain François and Hadley Wickham*: C++ and Rcpp for beginners
- *Markus Gesmann and Diego de Castillo*: Interactive web graphics with R and googleVis
- *Garrett Grolemund*: Data visualization with ggplot2
- *Stephanie A. Kovalchik*: Performing Meta-Analysis with R
- *Mark van der Loo and Edwin de Jonge*: An introduction to data cleaning with R
- *Martin Morgan*: R/Bioconductor for Analysis and Comprehension of High-throughput Genomic Data
- *George Ostrouchov and Drew Schmidt*: Programming with Big Data in R
- *Xavier de Pedro*: Web 2.0 interfaces for R with Tiki
- *Havard Rue*: Bayesian computing with INLA: An introduction item
- *Roger Bivand*: Using Spatial Data
- *Karim Chine*: R and Cloud Computing for Higher Education and Research
- *Andrea Dessi, Enrico Branca, Federico Figus*: Applied Financial Analysis and Human Capital Risk Management
- *Marco Scutari*: Learning Bayesian Networks in R: an Example in Systems Biology
- *Max Kuhn*: Predictive Modeling with R and the caret Package
- *Rui Paulo and Jesús Palomo*: Statistical Analysis of Computer Models using R
- *Josh Paulson and Joe Cheng*: Developing web applications with R and shiny

- *Pete Philipson*: Joint Modelling of Repeated Measurements and Time-to-Event Data
- *Andy South*: Making beautiful world maps with country-referenced data using rworldmap and other R packages
- *Tobias Verbeke and Stephan Wahlbrink*: Eclipse/StatET and Architect for Professional R Development
- *Alex Zolotovitski*: How to work with large R projects

Data analysis contest

We are pleased to announce the Data Analysis Contest for useR! 2013 attendants: http://www.edii.uclm.es/~useR-2013/#contest. Check the rules, download the data and send your proposal to win the prize!

Location & surrounding area

The University of Castilla-La Mancha is the only university in the region and it is divided into several campuses. The conference will take place at the campus in Albacete. Information and useful links on the numerous and famous attractions in the area surrounding the University of Castilla-La Mancha in Albacete can be found at the Castilla-La Mancha Tourism website http://www.visitclm.com/.

Further information

A web page offering more information on *useR! 2013*, including details regarding registration, is available at http://www.r-project.org/useR-2013/

We hope to meet you in Albacete!

Esteban Alfaro-Cortés, José Luis Alfaro-Navarro, María Teresa Alonso-Martínez, Emilio L. Cano, Gonzalo García-Donato, Matías Gámez-Martínez, Noelia García-Rubio, Virgilio Gómez-Rubio and Francisco Parreño-Torres.

The organizing committee, useR-2013@R-project.org

News from the Bioconductor Project

Bioconductor Team
Program in Computational Biology
Fred Hutchinson Cancer Research Center

Bioconductor 2.12 was released on 3 October 2012. It is compatible with R 3.0.1, and consists of 671 software packages and more than 675 up-to-date annotation packages. The release includes 65 new software packages, and enhancements to many others. Descriptions of new packages and updated NEWS files provided by current package maintainers are at http://bioconductor.org/news/bioc_2_12_release/.

Start using Bioconductor and R version 3.0.1 with

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

Install additional packages, e.g., **VariantTools**, with

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("VariantTools")
```

Upgrade installed packages with

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

New this release is `biocValid()`, a companion function to detect R / Bioconductor version mis-matches.

Explore available Bioconductor packages at http://bioconductor.org/packages/release/. All packages are grouped by 'BiocViews' to identify coherent groups of packages. Each package has an html page with the descriptions and links to vignettes, reference manuals, and use statistics.

A Bioconductor Amazon Machine Instance is available and updated; see http://bioconductor.org/help/bioconductor-cloud-ami.

Core annotation and software packages

This release includes **AnnotationHub**, a resource that enables ready access to large genome-scale resources (e.g., GTF or FASTA files from Ensembl; ENCODE tracks from UCSC) in formats (e.g., `GRanges` or `VCF` instances) that allow smooth integration with R work flows. The **AnnotationHub** resource can be queried through simple tab completion, or via metadata about resource provenance. Additional new annotation resources include **ensemblVEP** to query the Ensembl Variant Effect Predictor, and **KEGGREST** and **UniProt.ws** packages for on-line integration of data from corresponding resources. Our large collection of microarray- and organism-specific annotation packages have been updated to include current information.

GenomicRanges and related packages, e.g., **VariantAnnotation**, **IRanges**, **Biostrings**, **Rsamtools**, **GenomicFeatures** provide an extensive, mature and extensible framework for interacting with high throughput sequence data, either as a user or package developer. Many contributed packages rely on this infrastructure for interoperable, re-usable analysis.

Other activities

Bioconductor's Annual Meeting is in Seattle, 17-19 July 2013, see http://bioconductor.org/bioc2013; our European developer community meets in December, with final arrangements

pending. Additional training and community activities advertised at http://bioconductor.org/help/events/. The active Bioconductor mailing lists (http://bioconductor.org/help/mailing-list/) connect users with each other, to domain experts, and to maintainers eager to ensure that their packages satisfy the needs of leading edge approaches. Keep abreast of packages added to the 'devel' branch and other activities by following Bioconductor on Twitter.

Package developers will be interested in ongoing efforts planned for our next release. Activities include better integration of parallel evaluation, graphical interfaces to Bioconductor objects (e.g., via **shiny**, elaboration of **AnnotationHub** to support user-contributed and locally curated data, a repository for workflow and other packages that change infrequently but require significant computational resources to build, and efforts to ease integrate with github and other social coding resources. Our Google Summer of Code participants are enabling progress on some of these topics.

R Foundation News

by Kurt Hornik

Donations and new members

New benefactors

Quartz Bio, Switzerland

New supporting institutions

Institute for Geoinformatics, Westfälische Wilhelms-Universität Münster, Germany

New supporting members

Nicoleta Caragea, Romania
Alexandru Antoniade Ciprian, Romania
Martin Elff, Germany
Hubert Eser, Austria
Sarah "Mason" Garrison, USA
Christian A. Oberst, Germany

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Changes in R

From 2.15.3 to 3.0.1

by the R Core Team

CHANGES IN R 3.0.1

NEW FEATURES

- `chooseCRANmirror()` and `chooseBioCmirror()` gain an `ind` argument (like `setRepositories()`).
- `mcparallel` has a new argument `mc.interactive` which can modify the interactive flag in the child process. The new default is `FALSE` which makes child processes non-interactive by default (this prevents lock-ups due to children waiting for interactive input).
- `scan()` now warns when end-of-file occurs within a quoted string.
- `count.fields()` is now consistent with `scan()` in its handling of newlines in quoted strings. Instead of triggering an error, this results in the current line receiving `NA` as the field count, with the next line getting the total count of the two lines.
- The default method of `image()` will plot axes of the class of `xlim` and `ylim` (and hence of `x` and `y` if there is a suitable `range()` method). Based on a suggestion of Michael Sumner.
- `load()` now has a `verbose` argument for debugging support, to print the names of objects just before loading them.
- When loading a serialized object encounters a reference to a namespace which cannot be loaded, this is replaced by a reference to the global environment, with a warning.
- `pairs()` gains a `line.main` option for title placement.
- The remaining instances in which serialization to a raw vector was limited to 2GB have been unlimited on a 64-bit platform, and in most cases serialization to a vector of more than 1GB will be substantially faster.

UTILITIES

- `R CMD config` now make use of personal 'Makevars' files under '~/.R' and a site file 'Makevars.site', in the same way as `R CMD SHLIB` and `R CMD INSTALL`. This makes the utility more useful in package `configure` scripts.

  On Windows finding the personal files may require the environment variable `HOME` set.

  The old behaviour can be obtained with the new options '`--no-user-files`' and '`--no-site-files`'.

PACKAGE INSTALLATION

- Alternatives to the site and user customization files 'Makevars.site' and '~/.R/Makevars' can be specified *via* the environment variables `R_MAKEVARS_SITE` and `R_MAKEVARS_USER` respectively. These can be used to suppress the use of the default files by setting an empty value (where possible) or a non-existent path.

BUG FIXES

- `sys.source()` did not report error locations when `keep.source = TRUE`.
- `as.POSIXct.numeric` was coercing `origin` using the `tz` argument and not `"GMT"` as documented (PR#14973).
- `str(d)` no longer gives an error when `names(d)` contain illegal multibyte strings (PR#15247).
- Profiling of built-in functions with `line.profiling= TRUE` did not record the line from which they were called.
- `citation(pkg)` dropped the header and footer specified in the 'CITATION' file (PR#15257).
- Quotes were handled differently when reading the first line and reading the rest, so `read.table()` misread some files that contained quote characters (PR#15245).
- `cat()` with `sep` a character vector of length greater than one and more than one argument was using separators inconsistently (PR#15261).
- On Windows in R 3.0.0, `savePlot()` failed because of an incorrect check on the argument count.
- `unzip(list = TRUE)` returned `Names` as a factor and not a character vector (as documented) for the internal method. (Noticed by Sean O'Riordain.)
- `contourLines()` now checks more comprehensively for conformance of its `x`, `y` and `z` arguments (it was used incorrectly in package **R2G2**).
- Saved graphics display lists are R version-specific. Attempting to load workspaces containing them (or some other version-specific objects) aborted the load in R 3.0.0 and earlier; now it does a partial load and generates a warning instead.
- In R 3.0.0, `identify()` and `locator()` did not record information correctly, so replaying a graph (e.g. by copying it to another device) would fail. (PR#15271)
- Calling `file.copy()` or `dirname()` with the invalid input `""` (which was being used in packages, despite not being a file path) could have caused a segfault.

  `dirname("")` is now `""` rather than `"."` (unless it segfaulted).
- `supsmu()` could read/write outside its input vectors for very short inputs (seen in package **rms** for `n = 4`).
- `as.dendrogram()`'s `hclust` method uses less memory and hence gets considerably faster for large (n ~ 1000) clusterings, thanks to Daniel Müllner. (PR#15174)
- The return value when all workers failed from `parallel::mclapply(mc.preschedule = TRUE)` was a list of strings and not of error objects. (Spotted by Karl Forner and Bernd Bischl.)
- In R 3.0.0, when `help()` found multiple pages with the same alias, the HTML display of all the selections was not produced. (PR#15282)
- `splinefun(method="monoH.FC")` now produces a function with first argument named `x` and allows `deriv=3`, as documented. (PR#15273)
- `summaryRprof()` would only read the first `chunksize` lines of an `Rprof` file produced with `line.profiling=TRUE`. By default, this is the first 100 seconds. (PR#15288)
- `lsfit()` produced an incorrect error message when argument `x` had more columns than rows or `x` had a different number of rows than `y`. (Spotted by Renaud Gaujoux.)

- Binary operations on equal length vectors copied the class name from the second operand when the first had no class name, but did not set the object bit. (PR#15299)
- `write.table()` did not check that factors were constructed correctly, and so caused a segment fault when writing bad ones. (PR#15300)
- The internal HTTP server no longer chokes on POST requests without body. It will also pass-through other request types for custom handlers (with the method stored in Request-Method header) instead of failing.

CHANGES IN R 3.0.0

SIGNIFICANT USER-VISIBLE CHANGES

- Packages need to be (re-)installed under this version (3.0.0) of R.
- There is a subtle change in behaviour for numeric index values 2^{31} and larger. These never used to be legitimate and so were treated as `NA`, sometimes with a warning. They are now legal for long vectors so there is no longer a warning, and `x[2^31] <-y` will now extend the vector on a 64-bit platform and give an error on a 32-bit one.
- It is now possible for 64-bit builds to allocate amounts of memory limited only by the OS. It may be wise to use OS facilities (e.g. `ulimit` in a `bash` shell, `limit` in `csh`), to set limits on overall memory consumption of an R process, particularly in a multi-user environment. A number of packages need a limit of at least 4GB of virtual memory to load.

  64-bit Windows builds of R are by default limited in memory usage to the amount of RAM installed: this limit can be changed by command-line option '`--max-mem-size`' or setting environment variable `R_MAX_MEM_SIZE`.
- Negative numbers for colours are consistently an error: previously they were sometimes taken as transparent, sometimes mapped into the current palette and sometimes an error.

NEW FEATURES

- `identical()` has a new argument, `ignore.environment`, used when comparing functions (with default `FALSE` as before).
- There is a new option, `options(CBoundsCheck=)`, which controls how `.C()` and `.Fortran()` pass arguments to compiled code. If true (which can be enabled by setting the environment variable `R_C_BOUNDS_CHECK` to '`yes`'), raw, integer, double and complex arguments are always copied, and checked for writing off either end of the array on return from the compiled code (when a second copy is made). This also checks individual elements of character vectors passed to `.C()`.

  This is not intended for routine use, but can be very helpful in finding segfaults in package code.
- In `layout()`, the limits on the grid size have been raised (again).
- New simple `provideDimnames()` utility function.
- Where methods for `length()` return a double value which is representable as an integer (as often happens for package **Matrix**), this is converted to an integer.
- Matrix indexing of dataframes by two-column numeric indices is now supported for replacement as well as extraction.
- `setNames()` now has a default for its `object` argument, useful for a character result.

- `StructTS()` has a revised additive constant in the `loglik` component of the result: the previous definition is returned as the `loglik0` component. However, the help page has always warned of a lack of comparability of log-likelihoods for non-stationary models. (Suggested by Jouni Helske.)
- The logic in `aggregate.formula()` has been revised. It is now possible to use a formula stored in a variable; previously, it had to be given explicitly in the function call.
- `install.packages()` has a new argument `quiet` to reduce the amount of output shown.
- Setting an element of the graphics argument `lwd` to a negative or infinite value is now an error. Lines corresponding to elements with values `NA` or `NaN` are silently omitted.

  Previously the behaviour was device-dependent.
- Setting graphical parameters `cex`, `col`, `lty`, `lwd` and `pch` in `par()` now requires a length-one argument. Previously some silently took the first element of a longer vector, but not always when documented to do so.
- `Sys.which()` when used with inputs which would be unsafe in a shell (e.g. absolute paths containing spaces) now uses appropriate quoting.
- `as.tclObj()` has been extended to handle raw vectors. Previously, it only worked in the other direction. (Contributed by Charlie Friedemann, PR#14939.)
- New functions `cite()` and `citeNatbib()` have been added, to allow generation of in-text citations from `"bibentry"` objects. A `cite()` function may be added to `bibstyle()` environments.
- A `sort()` method has been added for `"bibentry"` objects.
- The `bibstyle()` function now defaults to setting the default bibliography style. The `getBibstyle()` function has been added to report the name of the current default style.
- `scatter.smooth()` now has an argument `lpars` to pass arguments to `lines()`.
- `pairs()` has a new `log` argument, to allow some or all variables to be plotted on logarithmic scale. (In part, wish of PR#14919.)
- `split()` gains a `sep` argument.
- `termplot()` does a better job when given a model with interactions (and no longer attempts to plot interaction terms).
- The parser now incorporates code from Romain Francois' **parser** package, to support more detailed computation on the code, such as syntax highlighting, comment-based documentation, etc. Functions `getParseData()` and `getParseText()` access the data.
- There is a new function `rep_len()` analogous to `rep.int()` for when speed is required (and names are not).
- The undocumented use `rep(NULL,length.out = n)` for `n >0` (which returns `NULL`) now gives a warning.
- `demo()` gains an `encoding` argument for those packages with non-ASCII demos: it defaults to the package encoding where there is one.
- `strwrap()` converts inputs with a marked encoding to the current locale: previously it made some attempt to pass through as bytes inputs invalid in the current locale.
- Specifying both `rate` and `scale` to `[dpqr]gamma` is a warning (if they are essentially the same value) or an error.

- `merge()` works in more cases where the data frames include matrices. (Wish of PR#14974.)
- `optimize()` and `uniroot()` no longer use a shared parameter object across calls. (`nlm()`, `nlminb()` and `optim()` with numerical derivatives still do, as documented.)
- The `all.equal()` method for date-times is now documented: times are regarded as equal (by default) if they differ by up to 1 msec.
- `duplicated()` and `unique()` gain a `nmax` argument which can be used to make them much more efficient when it is known that there are only a small number of unique entries. This is done automatically for factors.
- Functions `rbinom()`, `rgeom()`, `rhyper()`, `rpois()`, `rnbinom()`, `rsignrank()` and `rwilcox()` now return integer (not double) vectors. This halves the storage requirements for large simulations.
- `sort()`, `sort.int()` and `sort.list()` now use radix sorting for factors of less than 100,000 levels when `method` is not supplied. So does `order()` if called with a single factor, unless `na.last = NA`.
- `diag()` as used to generate a diagonal matrix has been re-written in C for speed and less memory usage. It now forces the result to be numeric in the case `diag(x)` since it is said to have 'zero off-diagonal entries'.
- `backsolve()` (and `forwardsolve()`) are now internal functions, for speed and support for large matrices.
- More matrix algebra functions (e.g. `chol()` and `solve()`) accept logical matrices (and coerce to numeric).
- `sample.int()` has some support for $n \geq 2^{31}$: see its help for the limitations.

  A different algorithm is used for `(n,size,replace = FALSE,prob = NULL)` for `n >1e7` and `size <= n/2`. This is much faster and uses less memory, but does give different results.
- `approxfun()` and `splinefun()` now return a wrapper to an internal function in the **stats** namespace rather than a `.C()` or `.Call()` call. This is more likely to work if the function is saved and used in a different session.
- The functions `.C()`, `.Call()`, `.External()` and `.Fortran()` now give an error (rather than a warning) if called with a named first argument.
- `Sweave()` by default now reports the locations in the source file(s) of each chunk.
- `clearPushBack()` is now a documented interface to a long-existing internal call.
- `aspell()` gains filters for R code, Debian Control Format and message catalog files, and support for R level dictionaries. In addition, package **utils** now provides functions `aspell_package_R_files()` and `aspell_package_C_files()` for spell checking R and C level message strings in packages.
- `bibentry()` gains some support for "incomplete" entries with a '`crossref`' field.
- `gray()` and `gray.colors()` finally allow `alpha` to be specified.
- `monthplot()` gains parameters to control the look of the reference lines. (Suggestion of Ian McLeod.)
- Added support for new `%~%` relation ("is distributed as") in plotmath.
- `domain = NA` is accepted by `gettext()` and `ngettext()`, analogously to `stop()` etc.

- `termplot()` gains a new argument `plot = FALSE` which returns information to allow the plots to be modified for use as part of other plots, but does not plot them. (Contributed by Terry Therneau, PR#15076.)
- `quartz.save()`, formerly an undocumented part of `R.app`, is now available to copy a device to a `quartz()` device. `dev.copy2pdf()` optionally does this for PDF output: `quartz.save()` defaults to PNG.
- The default method of `pairs()` now allows `text.panel = NULL` and the use of `<foo>.panel = NULL` is now documented.
- `setRefClass()` and `getRefClass()` now return class generator functions, similar to `setClass()`, but still with the reference fields and methods as before (suggestion of Romain Francois).
- New functions `bitwNot()`, `bitwAnd()`, `bitwOr()` and `bitwXor()`, using the internal interfaces previously used for classes `"octmode"` and `"hexmode"`.

  Also `bitwShiftL()` and `bitwShiftR()` for shifting bits in elements of integer vectors.
- New option `"deparse.cutoff"` to control the deparsing of language objects such as calls and formulae when printing. (Suggested by a comment of Sarah Goslee.)
- `colors()` gains an argument `distinct`.
- New `demo(colors)` and `demo(hclColors)`, with utility functions.
- `list.files()` (aka `dir()`) gains a new optional argument `no..` which allows to exclude `"."` and `".."` from listings.
- Multiple time series are also of class `"matrix"`; consequently, `head()`, e.g., is more useful.
- `encodeString()` preserves UTF-8 marked encodings. Thus if factor levels are marked as UTF-8 an attempt is made to print them in UTF-8 in `RGui` on Windows.
- `readLines()` and `scan()` (and hence `read.table()`) in a UTF-8 locale now discard a UTF-8 byte-order-mark (BOM). Such BOMs are allowed but not recommended by the Unicode Standard: however Microsoft applications can produce them and so they are sometimes found on websites.

  The encoding name `"UTF-8-BOM"` for a connection will ensure that a UTF-8 BOM is discarded.
- `mapply(FUN,a1,..)` now also works when `a1` (or a further such argument) needs a `length()` method (which the documented arguments never do). (Requested by Hervé Pagès; with a patch.)
- `.onDetach()` is supported as an alternative to `.Last.lib`. Unlike `.Last.lib`, this does not need to be exported from the package's namespace.
- The `srcfile` argument to `parse()` may now be a character string, to be used in error messages.
- The `format()` method for `ftable` objects gains a `method` argument, propagated to `write.ftable()` and `print()`, allowing more compact output, notably for LaTeX formatting, thanks to Marius Hofert.
- The `utils::process.events()` function has been added to trigger immediate event handling.
- `Sys.which()` now returns `NA` (not `""`) for `NA` inputs (related to PR#15147).
- The `print()` method for class `"htest"` gives fewer trailing spaces (wish of PR#15124).

  Also print output from `HoltWinters()`, `nls()` and others.

- `loadNamespace()` allows a version specification to be given, and this is used to check version specifications given in the '`Imports`' field when a namespace is loaded.
- `setClass()` has a new argument, `slots`, clearer and less ambiguous than `representation`. It is recommended for future code, but should be back-compatible. At the same time, the allowed slot specification is slightly more general. See the documentation for details.
- `mget()` now has a default for `envir` (the frame from which it is called), for consistency with `get()` and `assign()`.
- `close()` now returns an integer status where available, invisibly. (Wish of PR#15088.)
- The internal method of `tar()` can now store paths too long for the '`ustar`' format, using the (widely supported) GNU extension. It can also store long link names, but these are much less widely supported. There is support for larger files, up to the '`ustar`' limit of 8GB.
- Local reference classes have been added to package **methods**. These are a technique for avoiding unneeded copying of large components of objects while retaining standard R functional behavior. See `?LocalReferenceClasses`.
- `untar()` has a new argument `restore_times` which if false (not the default) discards the times in the tarball. This is useful if they are incorrect (some tarballs submitted to CRAN have times in a local timezone or many years in the past even though the standard required them to be in UTC).
- `replayplot()` cannot (and will not attempt to) replay plots recorded under R < 3.0.0. It may crash the R session if an attempt is made to replay plots created in a different build of R >= 3.0.0.
- Palette changes get recorded on the display list, so replaying plots (including when resizing screen devices and using `dev.copy()`) will work better when the palette is changed during a plot.
- `chol(pivot = TRUE)` now defaults to LAPACK, not LINPACK.
- The `parse()` function has a new parameter `keep.source`, which defaults to `options("keep.source")`.
- Profiling via `Rprof()` now optionally records information at the statement level, not just the function level.
- The `Rprof()` function now quotes function names in in its output file on Windows, to be consistent with the quoting in Unix.
- Profiling via `Rprof()` now optionally records information about time spent in GC.
- The HTML help page for a package now displays non-vignette documentation files in a more accessible format.
- To support `options(stringsAsFactors = FALSE)`, `model.frame()`, `model.matrix()` and `replications()` now automatically convert character vectors to factors without a warning.
- The `print` method for objects of class `"table"` now detects tables with 0-extents and prints the results as, e.g., '`<table of extent 0 x 1 x 2 >`'. (Wish of PR#15198.)
- Deparsing involving calls to anonymous functions has been made closer to reversible by the addition of extra parentheses.
- The function `utils::packageName()` has been added as a lightweight version of `methods::getPackageName()`.

- `find.package(lib.loc = NULL)` now treats loaded namespaces preferentially in the same way as attached packages have been for a long time.
- In Windows, the Change Directory dialog now defaults to the current working directory, rather than to the last directory chosen in that dialog.
- `available.packages()` gains a `"license/restricts_use"` filter which retains only packages for which installation can proceed solely based on packages which are guaranteed not to restrict use.
- New `check_packages_in_dir()` function in package **tools** for conveniently checking source packages along with their reverse dependencies.
- R's completion mechanism has been improved to handle help requests (starting with a question mark). In particular, help prefixes are now supported, as well as quoted help topics. To support this, completion inside quotes are now handled by R by default on all platforms.
- The memory manager now allows the strategy used to balance garbage collection and memory growth to be controlled by setting the environment variable `R_GC_MEM_GROW`. See `?Memory` for more details.
- ('For experts only', as the introductory manual says.) The use of environment variables `R_NSIZE` and `R_VSIZE` to control the initial (= minimum) garbage collection trigger for number of cons cels and size of heap has been restored: they can be overridden by the command-line options `--min-nsize` and `--min-vsize`; see `?Memory`.
- On Windows, the device name for bitmap devices as reported by `.Device` and `.Devices` no longer includes the file name. This is for consistency with other platforms and was requested by the **lattice** maintainer.

  `win.metafile()` still uses the file name: the exact form is used by package **tkrplot**.
- `set.seed(NULL)` re-initializes `.Random.seed` as done at the beginning of the session if not already set. (Suggestion of Bill Dunlap.)
- The `breaks` argument in `hist.default()` can now be a function that returns the breakpoints to be used (previously it could only return the suggested number of breakpoints).
- File 'share/licenses/licenses.db' has some clarifications, especially as to which variants of 'BSD' and 'MIT' is intended and how to apply them to packages. The problematic licence 'Artistic-1.0' has been removed.

LONG VECTORS

This section applies only to 64-bit platforms.

- There is support for vectors longer than $2^{31} - 1$ elements. This applies to raw, logical, integer, double, complex and character vectors, as well as lists. (Elements of character vectors remain limited to $2^{31} - 1$ bytes.)
- Most operations which can sensibly be done with long vectors work: others may return the error 'long vectors not supported yet'. Most of these are because they explicitly work with integer indices (e.g. `anyDuplicated()` and `match()`) or because other limits (e.g. of character strings or matrix dimensions) would be exceeded or the operations would be extremely slow.
- `length()` returns a double for long vectors, and lengths can be set to 2^{31} or more by the replacement function with a double value.

- Most aspects of indexing are available. Generally double-valued indices can be used to access elements beyond $2^{31} - 1$.
- There is some support for matrices and arrays with each dimension less than 2^{31} but total number of elements more than that. Only some aspects of matrix algebra work for such matrices, often taking a very long time. In other cases the underlying Fortran code has an unstated restriction (as was found for complex `svd()`).
- `dist()` can produce dissimilarity objects for more than 65536 rows (but for example `hclust()` cannot process such objects).
- `serialize()` to a raw vector is unlimited in size (except by resources).
- The C-level function `R_alloc` can now allocate 2^{35} or more bytes.
- `agrep()` and `grep()` will return double vectors of indices for long vector inputs.
- Many calls to `.C()` have been replaced by `.Call()` to allow long vectors to be supported (now or in the future). Regrettably several packages had copied the non-API `.C()` calls and so failed.
- `.C()` and `.Fortran()` do not accept long vector inputs. This is a precaution as it is very unlikely that existing code will have been written to handle long vectors (and the R wrappers often assume that `length(x)` is an integer).
- Most of the methods for `sort()` work for long vectors.

  `rank()`, `sort.list()` and `order()` support long vectors (slowly except for radix sorting).
- `sample()` can do uniform sampling from a long vector.

PERFORMANCE IMPROVEMENTS

- More use has been made of R objects representing registered entry points, which is more efficient as the address is provided by the loader once only when the package is loaded.

  This has been done for packages `base`, `methods`, `splines` and `tcltk`: it was already in place for the other standard packages.

  Since these entry points are always accessed by the R entry points they do not need to be in the load table which can be substantially smaller and hence searched faster. This does mean that `.C` / `.Fortran` / `.Call` calls copied from earlier versions of R may no longer work – but they were never part of the API.
- Many `.Call()` calls in package **base** have been migrated to `.Internal()` calls.
- `solve()` makes fewer copies, especially when `b` is a vector rather than a matrix.
- `eigen()` makes fewer copies if the input has dimnames.
- Most of the linear algebra functions make fewer copies when the input(s) are not double (e.g. integer or logical).
- A foreign function call (`.C()` etc) in a package without a `PACKAGE` argument will only look in the first DLL specified in the 'NAMESPACE' file of the package rather than searching all loaded DLLs. A few packages needed `PACKAGE` arguments added.
- The `@<-` operator is now implemented as a primitive, which should reduce some copying of objects when used. Note that the operator object must now be in package **base**: do not try to import it explicitly from package **methods**.

PACKAGE INSTALLATION

- The transitional support for installing packages without namespaces (required since R 2.14.0) has been removed. `R CMD build` will still add a namespace, but a `.First.lib()` function will need to be converted.

  `R CMD INSTALL` no longer adds a namespace (so installation will fail), and a `.First.lib()` function in a package will be ignored (with an installation warning for now).

  As an exception, packages without a 'R' directory and no 'NAMESPACE' file can still be installed.

- Packages can specify in their 'DESCRIPTION file' a line like

  ```
  Biarch: yes
  ```

  to be installed on Windows with '`--force-biarch`'.

- Package vignettes can now be processed by other engines besides `Sweave`; see 'Writing R Extensions' and the `tools::vignetteEngine` help topic for details.

- The '*.R' tangled source code for vignettes is now included in tarballs when `R CMD build` is used to produce them. In R 3.0.0, '*.R' files not in the sources will be produced at install time, but eventually this will be dropped.

- The package type "`mac.binary`" now looks in a path in the repository without any Mac subtype (which used to be '`universal`' or '`leopard`'): it looks in 'bin/macosx/contrib/3.0' rather than 'bin/macosx/leopard/contrib/2.15'). This is the type used for the CRAN binary distribution for OS X as from R 3.0.0.

- File 'etc/Makeconf' makes more use of the macros `$(CC)`, `$(CXX)`, `$(F77)` and `$(FC)`, so the compiler in use can be changed by setting just these (and if necessary the corresponding flags and `FLIBS`) in file '~/.R/Makevars'.

  This is convenient for those working with binary distributions of R, e.g. on OS X.

UTILITIES

- `R CMD check` now gives a warning rather than a note if it finds calls to `abort`, `assert` or `exit` in compiled code, and has been able to find the '.o' file in which the calls occur.

  Such calls can terminate the R process which loads the package.

- The location of the build and check environment files can now be specified by the environment variables `R_BUILD_ENVIRON` and `R_CHECK_ENVIRON`, respectively.

- `R CMD Sweave` gains a '`--compact`' option to control possibly reducing the size of the PDF file it creates when '`--pdf`' is given.

- `R CMD build` now omits Eclipse's '.metadata' directories, and `R CMD check` warns if it finds them.

- `R CMD check` now does some checks on functions defined within reference classes, including of `.Call()` etc calls.

- `R CMD check --as-cran` notes assignments to the global environment, calls to `data()` which load into the global environment, and calls to `attach()`.

- `R CMD build` by default uses the internal method of `tar()` to prepare the tarball. This is more likely to produce a tarball compatible with `R CMD INSTALL` and `R CMD check`: an external `tar` program, including options, can be specified *via* the environment variable `R_BUILD_TAR`.

- `tools::massageExamples()` is better protected against packages which re-define base functions such as `cat()` and `get()` and so can cause `R CMD check` to fail when checking examples.
- `R CMD javareconf` has been enhanced to be more similar to the code used by `configure`.

  There is now a test that a JNI program can be compiled (like `configure` did) and only working settings are used.

  It makes use of custom settings from configuration recorded in 'etc/javaconf'.
- The '`--no-vignettes`' argument of `R CMD build` has been renamed to the more accurate '`--no-build-vignettes`': its action has always been to (re)build vignettes and never omitted them.

  `R CMD check` accepts '`--no-build-vignettes`' as a preferred synonym for '`--no-rebuild-vignettes`'.

DEPRECATED AND DEFUNCT

- The `ENCODING` argument to `.C()` is defunct. Use `iconv()` instead.
- The `.Internal(eval.with.vis)` non-API function has been removed.
- Support for the converters for use with `.C()` has been removed, including the oft misused non-API header 'R_ext/RConverters.h'.
- The previously deprecated uses of `array()` with a 0-length `dim` argument and `tapply()` with a 0-length `INDEX` list are now errors.
- '`Translation`' packages are defunct.
- Calling `rep()` or `rep.int()` on a pairlist or other non-vector object is now an error.
- Several non-API entry points have been transferred to packages (e.g. `R_zeroin2`) or replaced by different non-API entry points (e.g. `R_tabulate`).
- The 'internal' graphics device invoked by `.Call("R_GD_nullDevice",package = "grDevices")` has been removed: use `pdf(file = NULL)` instead.
- The `.Fortran()` entry point `"dqrls"` which has not been used by R since version 2.15.1 is no longer available.
- Functions `traceOn()` and `traceOff()` in package **methods** are now defunct.
- Function `CRAN.packages()` is finally defunct.
- Use of `col2rgb(0)` is defunct: use `par("bg")` or `NA` instead.
- The long-defunct functions `Rd_parse()`, `anovalist.lm()`, `categpry()`, `clearNames()`, `gammaCody()`, `glm.fit.null()`, `lm.fit.null()`, `lm.wfit.null()`, `manglePackageNames()`, `mauchley.test()`, `package.contents()`, `print.coefmat()`, `reshapeLong()`, `reshapeWide()`, `tkclose()`, `tkcmd()`, `tkfile.dir()`, `tkfile.tail()`, `tkopen()`, `tkputs()`, `tkread()`, `trySilent()` and `zip.file.extract()` have been removed entirely (but are still documented in the help system).
- The unused `dataPath` argument to `attachNamespace()` has been removed.
- `grid.prompt()` has been removed: use `devAskNewPage()` instead.
- The long-deprecated `intensities` component is no longer returned by `hist()`.
- `mean()` for data frames and `sd()` for data frames and matrices are defunct.

- `chol(pivot = FALSE,LINPACK = TRUE)`, `ch2inv(LINPACK = TRUE)`, `eigen(EISPACK = TRUE)`, `solve(LINPACK = TRUE)` and `svd(LINPACK = TRUE)` are defunct: LAPACK will be used, with a warning.
- The `keep.source` argument to `library()` and `require()` is defunct. This option needs to be set at install time.
- Documentation for `real()`, `as.real()` and `is.real()` has been moved to 'defunct' and the functions removed.
- The `maxRasters` argument of `pdf()` (unused since R 2.14.0) has been removed.
- The unused `fontsmooth` argument has been removed from the `quartz()` device.
- All the (non-API) EISPACK entry points in R have been removed.
- `chol(pivot = TRUE,LINPACK = TRUE)` is deprecated.
- The long-deprecated use of `\synopsis` in the 'Usage' section of '.Rd' files will be removed in R 3.1.0.
- `.find.package()` and `.path.package()` are deprecated: only the public versions without the dot have ever been in the API.
- In a package's 'DESCRIPTION' file,

  ```
  License: X11
  ```

  is deprecated, since it includes 'Copyright (C) 1996 X Consortium' which cannot be appropriate for a current R package. Use 'MIT' or 'BSD_2_clause' instead.

CODE MIGRATION

- The C code underlying base graphics has been migrated to the **graphics** package (and hence no longer uses `.Internal()` calls).
- Most of the `.Internal()` calls used in the **stats** package have been migrated to C code in that package.

  This means that a number of `.Internal()` calls which have been used by packages no longer exist, including `.Internal(cor)` `.Internal(cov)`, `.Internal(optimhess)` and `.Internal(update.formula)`.
- Some `.External()` calls to the `base` package (really to the R executable or shared library) have been moved to more appropriate packages. Packages should not have been using such calls, but some did (mainly those used by `integrate()`).

PACKAGE parallel

- There is a new function `mcaffinity()` which allows getting or setting the CPU affinity mask for the current R process on systems that supports this (currently only Linux has been tested successfully). It has no effect on systems which do not support process affinity. Users are not expected to use this function directly (with the exception of fixing libraries that break affinity settings like OpenBLAS) – the function is rather intended to support affinity control in high-level parallel functions. In the future, R may supplement lack of affinity control in the OS by its own bookkeeping via `mcaffinity()` related to processes and threads it spawns.
- `mcparallel()` has a new argument `mc.affinity` which attempts to set the affinity of the child process according to the specification contained therein.

- The port used by socket clusters is chosen randomly: this should help to avoid clashes observed when two users of a multi-user machine try to create a cluster at the same time. To reproduce the previous behaviour set environment variable `R_PARALLEL_PORT` to `10187`.

C-LEVEL FACILITIES

- There has been some minor re-organization of the non-API header files. In particular, 'Rinternals.h' no longer includes the non-API header 'R_exts/PrtUtil.h', and that no longer includes 'R_exts/Print.h'.
- Passing `NULL` to `.C()` is now an error.
- `.C()` and `.Fortran()` now warn if `"single"` arguments are used with `DUP = FALSE`, as changes to such arguments are not returned to the caller.
- C entry points `R_qsort` and `R_qsort_I` now have `start` and `end` as `size_t` to allow them to work with longer vectors on 64-bit platforms. Code using them should be recompiled.
- A few recently added C entry points were missing the remapping to `Rf_`, notably `[dpq]nbinom_mu`.
- Some of the interface pointers formerly available only to `R.app` are now available to front-ends on all Unix-alikes: one has been added for the interface to `View()`.
- `PACKAGE = ""` is now an error in `.C()` etc calls: it was always contrary to the documentation.
- Entry point `rcont2` has been migrated to package **stats** and so is no longer available.
- `R_SVN_REVISION` in 'Rversion.h' is now an integer (rather than a string) and hence usable as e.g. `#if R_SVN_REVISION <70000`.
- The entry points `rgb2hsv` and `hsv2rgb` have been migrated to package **grDevices** and so are no longer available.
- `R_GE_version` has been increased to `10` and `name2col` removed (use `R_GE_str2col` instead). R internal colour codes are now defined using the typedef `rcolor`.
- The `REPROTECT` macro now checks that the protect index is valid.
- Several non-API entry points no longer used by R have been removed, including the Fortran entry points `chol`, `chol2inv`, `cg`, `ch` and `rg`, and the C entry points `Brent_fmin`, `fft_factor` and `fft_work`.
- If a `.External` call is registered with a number of arguments (other than `-1`), the number of arguments passed is checked for each call (as for other foreign function calls).
- It is now possible to write custom connection implementations outside core R using 'R_ext/Connections.h'. Please note that the implementation of connections is still considered internal and may change in the future (see the above file for details).

INTERNATIONALIZATION

- The management of translations has been converted to R code: see `?tools::update_pkg_po`.
- The translations for the R interpreter and `RGui.exe` are now part of the **base** package (rather than having sources in directory 'po' and being installed to 'share/locale'). Thus the **base** package supports three translation domains, `R-base`, `R` and `RGui`.

- The compiled translations which ship with R are all installed to the new package **translations** for easier updating. The first package of that name found on `.libPaths()` at the start of the R session will be used. (It is possible messages will be used before `.libPaths()` is set up in which case the default translations will be used: set environment variable `R_TRANSLATIONS` to point to the location of the intended **translations** package to use this right from the start.)
- The translations form a separate group in the Windows installer, so can be omitted if desired.
- The markup for many messages has been changed to make them easier to translate, incorporating suggestions from Łukasz Daniel.

INSTALLATION

- There is again support for building without using the C 'long double' type. This is required by C99, but system implementations can be slow or flawed. Use `configure` option '`--disable-long-double`'.
- `make pdf` and `make install-pdf` now make and install the full reference index (including all base and recommended packages).
- The 'reference manual' on the Windows GUI menu and included in the installer is now the full reference index, including all base and recommended packages.
- R help pages and manuals have no ISBNs because ISBN rules no longer allow constantly changing content to be assigned an ISBN.
- The Windows installer no longer installs a Start Menu link to the static help pages; as most pages are generated dynamically, this led to a lot of broken links.
- Any custom settings for Java configuration are recorded in file 'etc/javaconf' for subsequent use by `R CMD javareconf`.
- There is now support for `makeinfo` version 5.0 (which requires a slightly different '.texi' syntax).
- The minimum versions for '`--use-system-zlib`' and `--use-system-pcre` are now tested as 1.2.5 and 8.10 respectively.
- On Windows, the stack size is reduced to 16MB on 32-bit systems: misguided users were launching many threads without controlling the stack size.
- `configure` no longer looks for file '~/.Rconfig': '~/.R/config' has long been preferred.

BUG FIXES

- When `R CMD build` is run in an encoding other than the one specified in the package's 'DESCRIPTION' file it tries harder to expand the `authors@R` field in the specified encoding. (PR#14958)
- If `R CMD INSTALL` is required to expand the `authors@R` field of the 'DESCRIPTION' file, it tries harder to do so in the encoding specified for the package (rather than using ASCII escapes).
- Fix in package **grid** for pushing a viewport into a layout cell, where the layout is within a viewport that has zero physical width OR where the layout has zero total relative width (likewise for height). The layout column widths (or row heights) in this case were being calculated with non-finite values. (Reported by Winston Chang.)
- `solve(A,b)` for a vector `b` gave the answer names from `colnames(A)` for `LINPACK = TRUE` but not in the default case.

- `La.svd()` accepts logical matrices (as documented, and as `svd()` did).
- `legend()` now accepts negative `pch` values, in the same way `points()` long has.
- Parse errors when installing files now correctly display the name of the file containing the bad code.
- In Windows, tcltk windows were not always properly constructed. (PR#15150)
- The internal functions implementing `parse()`, `tools::parseLatex()` and `tools::parse_Rd()` were not reentrant, leading to errors in rare circumstances such as a garbage collection triggering a recursive call.
- Field assignments in reference class objects via `$<-` were not being checked because the magic incantation to turn methods on for that primitive operator had been inadvertently omitted.
- `setHook(hookname,value,action="replace")` set the hook to be the value, rather than a list containing the value as documented. (PR#15167)
- If a package used a 'NEWS.Rd' file, the main HTML package index page did not link to it. (Reported by Dirk Eddelbuettel.)
- The primitive implementation of `@<-` was not checking the class of the replacement. It now does a check, quicker but less general than `slot<-`. See the help.
- `split(x,f)` now recycles classed objects `x` in the same way as vectors. (Reported by Martin Morgan.)
- `pbeta(.28,1/2,2200,lower.tail=FALSE,log.p=TRUE)` is no longer `-Inf`; ditto for corresponding `pt()` and `pf()` calls, such as `pt(45,df=5000,lower.tail=FALSE,log.p=TRUE)`. (PR#15162)
- The Windows graphics device would crash R if a user attempted to load the graphics history from a variable that was not a saved history. (PR#15230)
- The workspace size for the `predict()` method for `loess()` could exceed the maximum integer size. (Reported by Hiroyuki Kawakatsu.)
- `ftable(x,row.vars,col.vars)` now also works when the `*.vars` arguments are (integer or character vectors) of length zero.
- Calling `cat()` on a malformed UTF-8 string could cause the Windows GUI to lock up. (PR#15227)
- `removeClass(cc)` gave "node stack overflow" for some class definitions containing `"array"` or `"matrix"`.

CHANGES IN R VERSION 2.15.3

NEW FEATURES

- `lgamma(x)` for very small `x` (in the denormalized range) is no longer `Inf` with a warning.
- `image()` now sorts an unsorted `breaks` vector, with a warning.
- The internal methods for `tar()` and `untar()` do a slightly more general job for 'ustar'-style handling of paths of more than 100 bytes.
- Packages **compiler** and **parallel** have been added to the reference index ('refman.pdf').

- `untar(tar = "internal")` has some support for `pax` headers as produced by e.g. `gnutar --posix` (which seems prevalent on OpenSUSE 12.2) or `bsdtar --format pax`, including long path and link names.
- `sQuote()` and `dQuote()` now handle 0-length inputs. (Suggestion of Ben Bolker.)
- `summaryRprof()` returns zero-row data frames rather than throw an error if no events are recorded, for consistency.
- The included version of PCRE has been updated to 8.32.
- The **tcltk** namespace can now be re-loaded after unloading.

  The Tcl/Tk event loop is inhibited in a forked child from package **parallel** (as in e.g. `mclapply()`).
- `parallel::makeCluster()` recognizes the value '`random`' for the environment variable `R_PARALLEL_PORT`: this chooses a random value for the port and reduces the chance of conflicts when multiple users start a cluster at the same time.

UTILITIES

- The default for `TAR` on Windows for `R CMD build` has been changed to be '`internal`' if no `tar` command is on the path.

  This enables most packages to be built 'out of the box' without `Rtools`: the main exceptions are those which need to be installed to re-build vignettes and need `Rtools` for installation (usually because they contain compiled code).

C-LEVEL FACILITIES

- On a 64-bit Windows platform with enough RAM, `R_alloc` can now allocate up to just under 32GB like other 64-bit platforms.

DEPRECATED AND DEFUNCT

- Use of `col2rgb(0)` is deprecated (see the help page for its limitations).
- The deprecated `intensities` component returned by `hist()` is no longer recognized by the `plot()` method and will be removed in R 3.0.0.
- `real()`, `as.real()` and `is.real()` are now formally deprecated and give a warning.
- This is formal notice that the non-API EISPACK entry points in R will be removed shortly.

INSTALLATION

- The `configure` tests for Objective C and Objective C++ now work on Mac OS 10.8 with Xcode 4.5.2 (PR#15107).
- The cairo-based versions of `X11()` now work with current versions of cairographics (e.g. 1.12.10). (PR#15168)

  A workaround for earlier versions of R is to use `X11.options(type = "nbcairo")`.
- Configuration and `R CMD javareconf` now come up with a smaller set of library paths for Java on Oracle-format JDK (including OpenJDK). This helps avoid conflicts between libraries (such as `libjpeg`) supplied in the JDK and system libraries. This can always be overridden if needed: see the 'R Installation and Administration' manual.

BUG FIXES

- `beta(a,b)` could overflow to infinity in its calculations when one of `a` and `b` was less than one. (PR#15075)
- `lbeta(a,b)` no longer gives `NaN` if `a` or `b` is very small (in the denormalized range).
- `bquote()` is now able to substitute default arguments in single-argument functions. (PR#15077)
- `browseEnv(html = FALSE)` would segfault if called from R (not `R.app`) on a CRAN-style Mac OS X build of R.
- '`[[<-`' for lists (generic vectors) needed to increment `NAMED` count when RHS is used more than once. (PR#15098)
- On Windows, warnings about opening a file or pipe with a non-ASCII description were sometimes output in UTF-8 rather than in the current locale's character set.
- The `call()` function did not duplicate its arguments. (PR#15115)
- `TukeyHSD()` could give `NA` results with some '`na.action`' methods such as `na.exclude()`. (Hinted at on R-help by John Fox.)
- The deprecated `svd(X,LINPACK = TRUE)` could alter `X` in R 2.15.[12]. (Reported by Bill Dunlap.)
- Under Windows, `file.link()` and `file.symlink()` used the link name twice, so would always fail. (Reported by Rui Barradas/Oliver Soong).
- `summaryRprof(memory = "both")` mixed up the units of Vcells and Ncells: it now works in bytes. (PR#15138)
- `tools::Rd2HTML()` would sometimes delete text. (PR#15134)
- `plot()` failed for `"table"` objects containing just one entry. (PR#15118)
- `embedFonts()` needed to quote some filepaths. (PR#15149)
- `parallel::mccollect()` handled `NULL` returns incorrectly (removing the element rather than setting it to `NULL`).
- The full reference index ('fullrefman.pdf') was missing packages **compiler** and **parallel**.
- The report for `optim(method = "L-BFGS-B",control = list(trace = 1))` reported the last completed and not the current iteration, unlike other methods and trace levels. (PR#15103)
- `qt(1e-12,1.2)` no longer gives `NaN`.
- `dt(1e160,1.2,log=TRUE)` no longer gives `-Inf`.
- On Windows the `untar()` function now quotes the directory name when using an external `tar` utility, so `R CMD check` will handle pathnames containing spaces.
- The version for Windows 8 and Windows Server 2012 is now displayed by `win.version()`. (Reported by Gabor Grothendieck.)
- The custom Windows installer target `myR` in the installer 'Makefile' did not work in 2.15.2. (Reported by Erich Neuwirth.)
- `aperm(matrix(1:6,2,dimnames=list(A={},B={})),"A")` no longer segfaults.
- Expressions involving user defined operators were not always deparsed faithfully. (PR#15179)

- The `enc2utf8()` function converted `NA_character_` to `"NA"` in non-UTF-8 locales. (PR#15201)
- The `exclude` argument to `xtabs()` was ignored for `"factor"` arguments.
- On Windows, work around an event-timing problem when the RGui console was closed from the 'X' control and the closure cancelled. (This would on some 64-bit systems crash R, typically those with a slow GPU relative to the CPU.)
- On unix `Rscript` will pass the `r_arch` setting it was compiled with on to the R process so that the architecture of `Rscript` and that of R will match unless overridden.
- On Windows, `basename()`, `dirname()` and `file.choose()` have more support for long non-ASCII file names with 260 or more bytes when expressed in UTF-8.

Changes on CRAN

2012-11-29 to 2013-05-25

by Kurt Hornik and Achim Zeileis

New CRAN task views

MetaAnalysis Topic: Meta-Analysis. Maintainer: Michael Dewey. Packages: **CRTSize**, **HSROC**, **MADAM**, **MAMA**, **MAc**, **MAd**, **MetABEL**, **MetaDE**, **MetaPCA**, **MetaPath**, **MetaQC**, **RcmdrPlugin.MA**, **SAMURAI**, **SCMA**, **bamdit**, **bspmma**, **compute.es**, **copas**, **epiR**, **gap**, **gemtc**, **mada**, **meta***, **metaLik**, **metaMA**, **metacor**, **metafor***, **metagen**, **metamisc**, **metatest**, **mvmeta**, **mvtmeta**, **psychometric**, **rmeta**, **selectMeta**, **skatMeta**.

SpatioTemporal Topic: Handling and Analyzing Spatio-Temporal Data. Maintainer: Edzer Pebesma. Packages: **GeoLight**, **M3**, **RNetCDF**, **RandomFields***, **RghcnV3**, **SpatioTemporal**, **Stem**, **adehabitatLT***, **argosfilter**, **cshapes**, **diveMove**, **googleVis**, **gstat***, **lgcp**, **lme4**, **mvtsplot**, **ncdf**, **ncdf4**, **nlme**, **openair**, **pastecs**, **pbdNCDF4**, **plm**, **plotKML**, **raster***, **rasterVis**, **rgl**, **solaR**, **sp***, **spBayes**, **spTimer**, **spacetime***, **spate**, **spatstat**, **sphet**, **splancs**, **splm**, **stam**, **stpp***, **stppResid**, **surveillance***, **trip***, **tripEstimation**, **xts***.

(* = core package)

New packages in CRAN task views

Bayesian **PAWL**, **RSGHB**, **bspec**, **eco**, **stochvol**.

ChemPhys **astro**, **simecol**, **stepPlr**.

ClinicalTrials **CRM**, **epibasix**.

Cluster **EMCluster**, **FisherEM**, **GLDEX**, **MFDA**, **Rmpi**, **latentnet**, **optpart**.

DifferentialEquations **PBSmodelling**, **primer**.

Distributions **ActuDistns***, **CDVine**, **Delaporte**, **GLDEX**, **PhaseType**, **VineCopula**, **copBasic**, **lmom**, **retimes**, **rlecuyer**.

Econometrics **LARF**, **RSGHB**, **partsm**, **survival**.

Environmetrics **earth**, **flexmix**, **fso**, **ipred**, **maptree**, **mda**, **metacom**, **primer**, **pvclust**, **quantreg**, **rioja**, **seas**, **surveillance**, **unmarked**, **untb**.

ExperimentalDesign **BatchExperiments**, **crossdes**, **displayHTS**, **gsbDesign**, **planor**.

Finance **BurStMisc**, **ESG**, **FinTS**, **GUIDE**, **PIN**, **SharpeR**, **highfrequency**, **nlme**, **parma**, **rmgarch**, **stochvol**.

Genetics **hierfstat**, **qtlbim**.

Graphics **RGtk2**, **ash**, **biclust**, **cba**, **diagram**, **igraph**, **onion**, **playwith**, **scagnostics**, **seriation**.

HighPerformanceComputing **bayesm**, **bigrf**, **doRNG**, **latentnet**, **mapReduce**, **mchof**, **pbdDEMO**, **pbdNCDF4**, **permGPU**, **rredis**, **sprint**, **xgrid**.

MachineLearning **Rmalschains**, **bigrf**, **frbs**, **maptree**.

MedicalImaging **bayesm**, **brainwaver**, **waveslim**.

Multivariate **FAiR**, **MFDA**, **cwhmisc**, **fso**.

NaturalLanguageProcessing **RcmdrPlugin.temis**, **SnowballC**, **qdap**, **tm.plugin.factiva**.

OfficialStatistics **JoSAE**, **MatchIt**, **SamplingStrata**, **hbsae**, **lavaan**, **lavaan.survey**, **lpSolve**, **odfWeave**, **rsae**, **samplingVarEst**, **tabplot**, **treemap**, **x12GUI**.

Optimization **adagio**.

Pharmacokinetics **PKPDmodels**, **deSolve**, **nlme**.

Phylogenetics **iteRates**.

Psychometrics **FAiR***, **MultiLCIRT**, **classify**, **fastICA**, **kst**, **lavaan.survey**, **mcIRT**, **pks**.

ReproducibleResearch **cacher**, **markdown**, **pander**, **rapport**.

Robust **RobLoxBioC**, **coxrobust**, **lqmm**, **robustX**, **robustlmm**, **rrcovHD**, **rrcovNA**.

SocialSciences **BMA**, **GPArotation**, **acepack**, **latentnet**, **lattice**, **lme4**, **meta**, **mvnmle**, **perturb**, **rgl**, **vcd**.

Spatial **DCluster***, **micromap**.

Survival **Biograph**, **SmoothHazard**, **SvyNom**, **aBioMarVsuit**, **bscr**, **bwsurvival**, **currentSurvival**, **gems**, **ipdmeta**, **jackknifeKME**, **lbiassurv**, **plsRcox**, **randomForestSRC**, **survexp.fr**, **timeROC**.

TimeSeries **FGN**, **MAR1**, **PVAClone**, **Quandl**, **TSTutorial**, **TimeProjection**, **arfima**, **brainwaver**, **bspec**, **events**, **fpp**, **nets**, **partsm**, **perARMA**, **rdatamarket**, **rts**, **seas**, **stochvol**, **surveillance**, **tframe**, **tsbugs**.

gR **BRugs**, **GeneNet**, **dclone**, **gRapHD**, **gRim**, **network**, **parcor**.

(* = core package)

New contributed packages

A3 Accurate, Adaptable, and Accessible Error Metrics for Predictive Models. Author: Scott Fortmann-Roe.

ABCExtremes ABC Extremes. Author: Rob Erhardt.

ABCp2 Approximate Bayesian Computational model for estimating P2. Authors: M. Catherine Duryea, Andrew D. Kern, Robert M. Cox, and Ryan Calsbeek.

ALDqr Algorithm Laplace density quantile regression. Authors: Luis Benites Sanchez, Victor Lachos.

ALKr Generate Age-Length Keys for fish populations. Authors: Jose Francisco Loff, Alberto Murta, Laurence Kell.

ARAMIS A R Adaptive Multiple Importance Sampling. Authors: Luca Pozzi, Antonietta Mira.

AnthropMMD A GUI for Mean Measures of Divergence. Author: Frederic Santos.

ArrayBin My First Collection of Functions. Author: Ed Curry.

BACprior Sensitivity of the Bayesian Adjustment for Confounding (BAC) algorithm to the choice of hyperparameter omega. Authors: Denis jf Talbot, Geneviève Lefebvre, Juli Atherton.

BH The Boost C++ Libraries. Authors: John W. Emerson, Michael J. Kane, Dirk Eddelbuettel, JJ Allaire, and Romain Francois.

BOG Bacterium and virus analysis of Orthologous Groups (BOG) is a package for identifying differentially regulated genes in the light of gene functions. Authors: Jincheol Park, Cenny Taslim, Shili Lin.

BSquare Bayesian Simultaneous Quantile Regression. Author: Luke Smith & Brian Reich.

BaySIC Bayesian Analysis of Significantly Mutated Genes in Cancer. Author: Nicholas B. Larson.

BayesBridge Bridge Regression. Authors: Nicholas G. Polson, James G. Scott, and Jesse Windle.

BayesComm Bayesian community ecology analysis. Author: Nick Golding.

BayesVarSel Bayesian Variable selection in Linear Models. Authors: Gonzalo Garcia-Donato and Anabel Forte.

BcDiag Diagnostics plots for Bicluster Data. Authors: Aregay Mengsteab, Martin Otava, Tatsiana Khamiakova.

BenfordTests Statistical Tests for Evaluating Conformity to Benford's Law. Authors: Dieter William Joenssen, with contributions from Thomas Muellerleile.

BiDimRegression Calculates the bidimensional regression between two 2D configurations. Author: Claus-Christian Carbon.

BlockMessage Creates strings that show a text message in 8 by 8 block letters. Authors: Elliot Noma, Aliona Manvae.

BurStMisc Burns Statistics miscellaneous. Author: Burns Statistics. In view: *Finance*.

CAMAN Finite Mixture Models and meta-analysis tools — based on C.A.MAN. Authors: Peter Schlattmann, Johannes Hoehne.

CCAGFA Bayesian canonical correlation analysis and group factor analysis. Authors: Seppo Virtanen and Arto Klami.

CDLasso Coordinate Descent Algorithms for Lasso Penalized L1, L2, and Logistic Regression. Authors: Edward Grant, Kenneth Lange, Tong Tong Wu.

CLAG An unsupervised non hierarchical clustering algorithm handling biological data. Authors: Linda Dib, Raphael Champeimont, Alessandra Carbone.

COBRA Nonlinear Aggregation of Predictors. Author: Benjamin Guedj.

CUMP Analyze Multivariate Phenotypes by Combining Univariate results. Authors: Xuan Liu and Qiong Yang.

CaDENCE Conditional Density Estimation Network Construction and Evaluation. Author: Alex J. Cannon.

CfEstimateQuantiles Estimate quantiles using any order Cornish-Fisher expansion. Author: Maxim Yurchuk.

CheckDigit Calculate and verify check digits. Author: Justin Brantley.

CoinMinD Simultaneous Confidence Interval for Multinomial Proportion. Author: M. Subbiah.

Comp2ROC Compare two ROC curves that intersect. Authors: Ana C. Braga, with contributions from Hugo Frade.

ConConPiWiFun An implementation of continuous convex piecewise (linear) functions. Author: Robin Girard.

CondReg Condition Number Regularized Covariance Estimation. Authors: Sang-Yun Oh, Bala Rajaratnam, Joong-Ho Won.

CountsEPPM Mean and variance modeling of count data. Authors: David M Smith, Malcolm J Faddy.

DMR Delete or Merge Regressors for linear model selection. Authors: Aleksandra Maj, Agnieszka Prochenka, Piotr Pokarowski.

DPw Semiparametric Bayesian procedure for selecting a subset containing the weakest species with an acceptably high probability. Author: Yumi Kondo.

DTComPair Comparison of Binary Diagnostic Tests in a Paired Study Design. Authors: Christian Stock, Thomas Hielscher.

DataCombine R tools for making it easier to combine and clean data sets. Author: Christopher Gandrud.

DataFrameConstr Constrained data frames and homogenous list classes. Author: Jeffrey Arnold.

Delaporte Statistical functions for the Delaporte distribution. Author: Avraham Adler. In view: *Distributions*.

Demerelate Functions to calculate relatedness on diploid genetic data. Authors: Philipp Kraemer and Gabriele Gerlach.

DendSer Dendrogram seriation: ordering for visualisation. Authors: Catherine B. Hurley and Denise Earle.

Digiroo2 An application programming interface for generating null models of social contacts based on individuals' space use. Authors: Ross Dwyer, Emily Best and Anne Goldizen.

Dominance ADI (average dominance index) and social network graphs with dual directions. Authors: Knut Krueger, with contributions from Konstanze Krueger.

EMCluster EM Algorithm for Model-Based Clustering of Finite Mixture Gaussian Distribution. Authors: Wei-Chen Chen [aut, cre], Ranjan Maitra [aut], Volodymyr Melnykov [aut]. In view: *Cluster*.

ESG Asset projection. Authors: Jean-Charles Croix, Thierry Moudiki, Frédéric Planchet, Wassim Youssef. In view: *Finance*.

EasyHTMLReport Easy to send HTML reports. Author: Yohei Sato.

EasyUpliftTree Easy Uplift Tree Model. Authors: Yohei Sato, Issei Kurahashi.

EpiModel Mathematical Modeling of Infectious Disease. Author: Samuel Jenness [cre, aut].

ExactCIdiff Inductive Confidence Intervals for the difference between two proportions. Authors: Guogen Shan, Weizhen Wang.

ExactNumCI Exact Confidence Interval for binomial proportions. Authors: Deqiang Sun, Hyun Jung Park.

ExactPath Exact solution paths for regularized LASSO regressions with L_1 penalty. Author: Kai Wang.

ExactSampling ExactSampling: risk evaluation using exact resampling methods for the k Nearest Neighbor algorithm. Author: Kai Li.

ExceedanceTools Confidence Regions for Exceedance Sets. Author: Joshua French.

ExpDes.pt Pacote Experimental Designs (Portuguese). Authors: Eric Batista Ferreira, Pórtya Piscitelli Cavalcanti, Denismar Alves Nogueira.

FAOSTAT A complementary package to the FAOSTAT database and the Statistical Yearbook of the Food and Agricultural Organization of the United Nations. Author: Michael C. J. Kao.

FBFsearch Algorithm for searching the space of Gaussian directed acyclic graphical models through moment fractional Bayes factors. Authors: Davide Altomare, Guido Consonni and Luca La Rocca.

FI Provide functions for forest inventory calculations. Author: David V. Dias [aut, cre].

FRAPO Financial Risk Modelling and Portfolio Optimisation with R. Authors: Bernhard Pfaff [aut, cre], Miguel Sousa Lobo [ctb] (SOCP), Lieven Vandenberge [ctb] (SOCP), Stephen Boyd [ctb] (SOCP), Herve Lebret [ctb] (SOCP).

FastPCS Compute the FastPCS outlyingness index. Author: Kaveh Vakili.

FindIt Find Heterogeneous Treatment Effects. Authors: Marc Ratkovic, Kosuke Imai.

FluOMatic Estimation of background-subtracted fluorescence data. Authors: Sebastien Joucla, Christophe Pouzat.

FuzzyNumbers Tools to deal with fuzzy numbers in R. Author: Marek Gagolewski.

FuzzyStatProb Fuzzy stationary probabilities from a sequence of observations of an unknown Markov chain. Author: Pablo J. Villacorta.

FuzzyToolkitUoN Type 1 Fuzzy Logic Toolkit. Authors: Craig Knott, Luke Hovell, Nathan Karimian with supervision from Jon Garibaldi.

GAIPE Graphical Extension with Accuracy in Parameter Estimation (GAIPE). Author: Tzu-Yao Lin.

GESTr Gene Expression State Transformation. Author: Ed Curry.

GOsummaries Word cloud summaries of GO enrichment analysis. Author: Raivo Kolde.

GRaF Species distribution modelling using latent Gaussian random fields. Author: Nick Golding.

GSIF Global Soil Information Facilities. Authors: Tomislav Hengl [cre, aut], Bas Kempen [aut], Gerard Heuvelink [aut], Brendan Malone [ctb], Hannes Reuter [ctb].

GUIDE GUI for DErivatives in R. Author: S Subramanian. In view: *Finance*.

GWG Calculation of probabilities for inadequate and excessive gestational weight gain. Author: Christina Riedel.

GeneticTools Collection of Genetic Data Analysis Tools. Author: Daniel Fischer.

GetR GetR: Calculate Guttman error trees in R. Authors: Johannes Beller, Soeren Kliem.

GibbsACOV Gibbs Sampler for One-Way Mixed-Effects ANOVA and ANCOVA Models. Authors: Emily Goren and Quan Zhang.

GoFKernel GoFKernel: Testing Goodness-of-fit with the Kernel Density Estimator. Author: Jose M. Pavia.

GriegSmith Uses Grieg-Smith method on 2 dimentional spatial data. Author: Brian McGuire. In view: *Spatial*.

GuardianR Guardian API Wrapper. Author: Marco Toledo Bastos & Cornelius Puschmann.

HAP.ROR Recursive Organizer (ROR). Authors: Lue Ping Zhao and Xin Huang.

HPOSim Analysis similarities between HPO terms and phenotypic similarity between genes and between diseases. Authors: Yue Deng, Gang Wang, Xiaocheng Huang and Tingting Ma.

IBDhaploRtools Functions for the Analysis of IBD Haplo output. Author: Marshall Brown.

IBHM Approximation using the IBHM method. Author: Pawel Zawistowski.

IM Orthogonal Moment Analysis. Authors: Bartek Rajwa, Murat Dundar, Allison Irvine, Tan Dang.

Iboot Iboot: iterated bootstrap tests and confidence sets. Author: Nicola Lunardon.

InferenceSMR Inference about the standardized mortality ratio when evaluating the effect of a screening program on survival. Authors: Denis Talbot, Thierry Duchesne, Jacques Brisson, Nathalie Vandal.

InteractiveIGraph Interactive network analysis and visualization. Author: Vygantas Butkus.

KappaGUI GUI for Cohen's and Fleiss' Kappa. Author: Frederic Santos.

Kmisc Miscellaneous functions intended to improve the R coding experience. Author: Kevin Ushey.

LARF Local Average Response Functions for Estimating Treatment Effects. Authors: Weihua An and Xuefu Wang. In view: *Econometrics*.

LDExplorer Efficient Whole-Genome LD Based Haplotpe Block Recognition. Authors: Daniel Taliun, Johann Gamper, Cristian Pattaro.

LDOD Finding Locally D-optimal optimal designs for some nonlinear and generalized linear models. Authors: Ehsan Masoudi, Majid Sarmad and Hooshang Talebi.

Lock5Data Datasets for "Statistics: UnLocking the Power of Data". Author: Robin Lock.

MALDIquantForeign Import/Export routines for **MALDIquant**. Author: Sebastian Gibb [aut, cre].

MAR1 Multivariate Autoregressive Modeling for Analysis of Community Time-Series Data. Author: Lindsay P. Scheef. In view: *TimeSeries*.

MASSTIMATE Body Mass Estimation Equations for Vertebrates. Author: Nicolas E. Campione.

MCPerm A Monte Carlo permutation method for multiple test correlation in case/control association study. Authors: Lanying Zhang and Yongshuai Jiang.

MDPtoolbox Markov Decision Processes toolbox. Authors: Iadine Chades, Guillaume Chapron, Marie-Josee Cros, Frederick Garcia, Regis Sabbadin.

MEET Motif Elements Estimation Toolkit. Authors: Joan Maynou and Erola Pairo.

MOJOV Mojo Variants: Rare Variants analysis. Author: Ke-Hao Wu.

MSwM Fitting Markov Switching Models. Authors: Josep A. Sanchez-Espigares, Alberto Lopez-Moreno.

MVN Multivariate Normality Tests. Author: Selcuk Korkmaz.

MapGAM Mapping Smoothed Odds Ratios from Individual-Level Data. Authors: Veronica Vieira, Scott Bartell, and Robin Bliss.

MedOr Median Ordering Statistical R package. Authors: Adriano Polpo, Carlos Alberto de Braganca Pereira.

MetaSKAT Meta analysis for SNP-set (Sequence) Kernel Association Test. Author: Seunggeun Lee.

MiClip A Model-based Approach to Identify Binding Sites in CLIP-Seq Data. Author: Tao Wang.

MiST Mixed effects Score Test for continuous outcomes. Authors: Jianping Sun, Yingye Zheng, and Li Hsu.

MicroStrategyR Author: Rick Pechter.

MixMAP Implements the MixMAP algorithm. Author: Gregory J. Matthews.

MixtureInf Inference for Finite Mixture Models. Authors: Jiahua Chen and Pengfei Li.

MonetDB.R Connect MonetDB to R. Authors: Hannes Muehleisen [aut, cre], Thomas Lumley [ctb], Anthony Damico [ctb].

MonoPoly Functions to fit monotone polynomials. Authors: Berwin A. Turlach [aut, cre], Kevin Murray [ctb].

Morpho Calculations and visualizations related to Geometric Morphometrics. Author: Stefan Schlager.

MuFiCokriging Multi-Fidelity Cokriging models. Author: Loic Le Gratiet.

MultinomialCI Simultaneous confidence intervals for multinomial proportions according to the method by Sison and Glaz. Author: Pablo J. Villacorta.

NHEMOtree Non-hierarchical evolutionary multi-objective tree learner to perform cost-sensitive classification. Author: Swaantje Casjens.

NPCD Nonparametric Methods for Cognitive Diagnosis. Authors: Yi Zheng and Chia-Yi Chiu, with contributions from Jeffrey A. Douglas.

NPCirc Nonparametric Circular Methods. Authors: María Oliveira, Rosa M. Crujeiras and Alberto Rodríguez-Casal.

Nozzle.R1 Nozzle Reports. Author: Nils Gehlenborg.

OPDOE OPtimal Design Of Experiments. Authors: Petr Simecek, Juergen Pilz Mingui Wang, Albrecht Gebhardt.

OmicKriging OmicKriging for Phenotypic Prediction. Authors: Hae Kyung Im, Heather E. Wheeler.

OneTwoSamples Deal with one and two (normal) samples. Author: Ying-Ying Zhang (Robert).

OpenMPController Control number of OpenMP threads dynamically. Author: Simon Guest.

OptInterim Optimal Two and Three Stage Designs for Single-Arm and Two-Arm Randomized Controlled Trials with a Long-Term Binary Endpoint. Authors: Bo Huang and Neal Thomas.

OrdLogReg Ordinal Logic Regression. Author: Bethany Wolf.

OutlierDC Outlier Detection using Quantile Regression for Censored Data. Authors: Soo-Heang Eo and HyungJun Cho.

PAGI Identify the dysregulated KEGG pathways based on global influence from the internal effect of pathways and crosstalk between pathways. Authors: Junwei Han, Yanjun Xu, Haixiu Yang, Chunquan Li and Xia Li.

PBSmapping Mapping Fisheries Data and Spatial Analysis Tools. Authors: Jon T. Schnute, Nicholas Boers, Rowan Haigh, Chris Grandin, Angus Johnson, Paul Wessel, Franklin Antonio. In view: *Spatial*.

PCovR Principal Covariates Regression. Authors: Marlies Vervloet [aut, cre], Henk Kiers [aut], Eva Ceulemans [ctb].

PIN Estimates the parameters of a trading-tree model for the computation of the probability of informed trading. Author: P. Zagaglia. In view: *Finance*.

PROTOLIDAR PRocess TOol LIdar DAta in R. Author: Monica Fernanda Rinaldi.

PReMiuM Dirichlet Process Bayesian Clustering, Profile Regression. Authors: David I. Hastie, Silvia Liverani and Sylvia Richardson, with a contribution by Lamiae Azizi.

PResiduals Probability scale residuals and residual correlations. Authors: Charles Dupont, Chun Li, Bryan Shepherd.

PST Probabilistic Suffix Trees. Author: Alexis Gabadinho.

PamGeneMixed Preprocessing and Modeling Kinase Activity Profiles in PamChip Data. Authors: Pushpike Thalikarathne, Ziv Shkedy, Dan Lin, Lieven Clement, and Geert Verbeke.

PlotRegionHighlighter Creates an envelope that surrounds a set of points plotted in a two dimensional space. Author: Elliot Noma.

PracTools Tools for Designing and Weighting Survey Samples. Authors: Richard Valliant, Jill A. Dever, Frauke Kreuter.

PropClust Propensity Clustering and Decomposition. Authors: John Michael O Ranola, Kenneth Lange, Steve Horvath, Peter Langfelder.

PropScrRand Propensity score methods for assigning treatment in randomized trials. Author: Travis M. Loux.

PurBayes Bayesian Estimation of Tumor Purity and Clonality. Author: Nicholas B. Larson.

Quandl Quandl Data Connection. Authors: Raymond McTaggart, Gergely Daroczi. In view: *TimeSeries*.

RAFM Admixture F-model. Author: Markku Karhunen.

RAP Reversal Association Pattern. Authors: U. Sangeetha, M. Subbiah with considerable contribution from M.R. Srinivasan.

RCA Relational Class Analysis. Authors: Amir Goldberg, Gabor Csardi, Jinjian Zhai.

RCircos Circos 2D Track Plot. Author: Hongen Zhang.

RClimMAWGEN R Climate Index Multi-site Auto-regressive Weather GENeretor: a package to generate time series of climate indices from RMAWGEN generations. Authors: Emanuele Cordano, Annalisa Di Piazza.

RDIDQ Perform Quality check on data. Author: Rahul Mehta.

RDSTK An R wrapper for the Data Science Toolkit API. Authors: Ryan Elmore and Andrew Heiss.

REBayes Empirical Bayes Estimation and Inference in R. Author: Roger Koenker.

RFGLS a family GWAS data-analysis tool that uses a generalized least squares method to perform single-marker association analysis. Authors: Xiang Li, Robert M. Kirkpatrick, and Saonli Basu.

RMessenger IM Client for R. Authors: Wush Wu, Jack Moffitt and Patrick Powell.

ROSE ROSE: Random Over-Sampling Examples. Authors: Nicola Lunardon, Giovanna Menardi, Nicola Torelli.

RSA Response surface analyses. Author: Felix Schönbrodt.

RSAP SAP Netweaver RFC connector for R. Author: Piers Harding.

RSGHB Functions for Hierarchical Bayesian Estimation: A Flexible Approach. Authors: Jeff Dumont, Jeff Keller, Chase Carpenter. In views: *Bayesian*, *Econometrics*.

RSiteCatalyst Adobe (Omniture) Reporting API. Author: Randy Zwitch & Jowanza Joseph.

RadialPlotter Statistical age models analysis in optically stimulated luminescence dating. Author: Peng Jun.

RbioRXN Process Rhea and MetaCyc (BioCyc, EcoCyc) biochemical reaction data. Authors: Byoungnam Min, Kyoung Heon Kim and In-Geol Choi.

RcmdrPlugin.EZR R Commander Plug-in for the EZR (Easy R) Package. Author: Yoshinobu Kanda.

RcmdrPlugin.MA Graphical User Interface for Conducting Meta-Analyses in R. Author: AC Del Re. In view: *MetaAnalysis*.

RcmdrPlugin.SM Rcmdr Sport Management Plug-In. Author: Stéphane Champely.

RcmdrPlugin.lfstat Rcmdr Plug-In for low flow analysis. Authors: Daniel Koffler and Gregor Laaha.

RcmdrPlugin.plotByGroup Rcmdr plots by group using lattice. Author: Poul Svante Eriksen with contributions by Ege Rubak.

RcmdrPlugin.seeg Rcmdr Plugin for **seeg**. Author: Miguel F. Acevedo.

RcmdrPlugin.temis Graphical user interface providing an integrated text mining solution. Authors: Milan Bouchet-Valat [aut, cre], Gilles Bastin [aut]. In view: *NaturalLanguageProcessing*.

RcppClassicExamples Examples using RcppClassic to interface R and C++. Authors: Dirk Eddelbuettel and Romain Francois, based on code written during 2005 and 2006 by Dominick Samperi.

RcppProgress An interruptible progress bar with OpenMP support for C++ in R packages. Author: Karl Forner.

RcppRoll Fast rolling functions through Rcpp and RcppArmadillo. Author: Kevin Ushey.

RcppXts Interface the xts API via Rcpp. Author: Dirk Eddelbuettel.

Reol R interface to the Encyclopedia of Life. Authors: Barb Banbury, Brian O'Meara.

Rgbp Bayesian Hierarchical Modeling and Frequentist Method Check. Authors: Joseph Kelly, Carl Morris, and Hyungsuk Tak.

Rgnuplot R interface for gnuplot. Authors: Göran Högnäs [ths], Nicolas Devillard [aut], Mauricio Galo [ctb], Patrick J. Bartlein [ctb], Oscar Perpiñán Lamigueiro [ctb], José Gama [aut, cre].

Ritc Isothermal Titration Calorimetry (ITC) Data Analysis. Author: Yingyun Liu.

SAMURAI Sensitivity Analysis of a Meta-analysis with Unpublished but Registered Analytical Investigations. Authors: Noory Y. Kim. Advisors: Shrikant I. Bangdiwala, Gerald Gartlehner. In view: *MetaAnalysis*.

SAVE Implementation of the SAVE approach for Computer Models. Authors: Rui Paulo, Gonzalo Garcia-Donato, and Jesus Palomo, with contributions from J. Berger, S. Bayarri and J. Sacks.

SDBP Calculate the third-order accurate Unbiased P-values via Speedy double bootstrap method. Author: Aizhen Ren.

SDD Serial Dependence Diagrams. Authors: Luca Bagnato, Lucio De Capitani, Angelo Mazza and Antonio Punzo.

SEMID Identifiability of linear structural equation models. Authors: Rina Foygel, Mathias Drton.

SLHD Maximin-Distance (Sliced) Latin Hypercube Designs. Author: Shan Ba.

SMCRM Data Sets for Statistical Methods in Customer Relationship Management by Kumar and Petersen (2012). Authors: Tobias Verbeke, based on datasets provided on the book's website.

SMFI5 R functions and data from Chapter 5 of "Statistical Methods for Financial Engineering". Author: Bruno Remillard.

SMNCensReg Fitting univariate censored regression model under the scale mixture of normal distributions. Authors: Aldo M. Garay, Victor Lachos and Monique Bettio Massuia.

SNPMClust A bivariate Gaussian genotype clustering and calling algorithm for Illumina microarrays. Authors: Stephen W. Erickson, with contributions from Joshua Callaway.

SNPtools Accessing, subsetting and plotting mouse SNPs. Author: Daniel Gatti.

SODC Optimal Discriminant Clustering (ODC) and Sparse Optimal Discriminant Clustering (SODC). Author: Yanhong Wang.

SOLOMON Parentage analysis. Author: Mark Christie.

SSDforR SSD for R to analyze single system data. Authors: Charles Auerbach, Wendy Zeitlin Schudrich.

SSN Spatial Modeling on Stream Networks. Authors: Jay Ver Hoef and Erin Peterson.

SUE Subsampling method. Author: Jim Yi.

SemiMarkov Multi-States Semi-Markov Models. Authors: Agnieszka Listwon, Philippe Saint-Pierre.

Sequential Exact Sequential Analysis for Poisson Data. Authors: Ivair Ramos Silva and Martin Kulldorff.

SetMethods SetMethods: A Package Companion to "Set-Theoretic Methods for the Social Sciences"" Author: Mario Quaranta.

SharpeR Statistical significance of Sharpe ratio. Author: Steven E. Pav. In view: *Finance*.

Skillings.Mack The Skillings-Mack test Statistic for block designs with missing observations. Authors: Patchanok Srisuradetchai, John J. Borkowski.

SmoothHazard Fitting illness-death model for interval-censored data. Authors: Celia Touraine, Pierre Joly, Thomas A. Gerds. In view: *Survival*.

SnowballC Snowball stemmers based on the C libstemmer UTF-8 library. Author: Milan Bouchet-Valat [aut, cre]. In view: *NaturalLanguageProcessing*.

SparseTSCGM Sparse time series chain graphical models. Authors: Fentaw Abegaz and Ernst Wit.

SphericalCubature Numerical integration over spheres and balls in n-dimensions; multivariate polar coordinates. Authors: John P. Nolan.

StVAR Student's t Vector Autoregression (StVAR). Author: Niraj Poudyal.

Stack Stylized concatenation of data.frames or ffdfs. Author: Mike Malecki.

StandardizeText Standardize Text. Author: David Nepomechie.

Stat2Data Datasets for Stat2. Author: Robin Lock.

SvyNom Nomograms for Right-Censored Outcomes from Survey Designs. Authors: Mithat Gonen, Marinela Capanu. In view: *Survival*.

TDD Time-Domain Deconvolution of Seismometer Response. Author: Jake Anderson.

TestScorer Scores tests, shows and saves the results. Author: Manel Salamero.

TimeMachine Time Machine. Authors: Gianluca Campanella [aut, cre], Marc Chadeau-Hyam [aut], Maria De Iorio [ctb], Ajay Jasra [ctb].

TimeProjection Time Projections. Author: Jeffrey Wong. In view: *TimeSeries*.

TraMineRextras Extras for use with the **TraMineR** package. Authors: Gilbert Ritschard, Reto Bürgin and Matthias Studer, with contributions from Alexis Gabadinho, Nicolas Müller and Patrick Rousset.

TreeSimGM Simulating Phylogenetic Trees under a General Model. Authors: Oskar Hagen, Tanja Stadler.

UWHAM Unbinned weighted histogram analysis method (UWHAM). Authors: Zhiqiang Tan and Emilio Gallicchio.

VGAMdata Data supporting the **VGAM** package. Author: Thomas W. Yee.

VisuClust Authors: Michael Sieger and Georg Ohmayer.

WARN Weaning Age Reconstruction with Nitrogen isotope analysis. Author: Takumi Tsutaya.

WebDevelopR Website development package for R. Authors: Evan Ray, Peter Krafft, John Staudenmayer.

WeightedCluster Clustering of Weighted Data. Author: Matthias Studer.

WordPools Classical word pools used in studies of learning and memory. Author: Michael Friendly.

XiMpLe A simple XML tree parser and generator. Author: m.eik michalke.

YplantQMC Plant architectural analysis with Yplant and QuasiMC. Authors: Remko Duursma. QuasiMC by Mik Cieslak. Uses code by Robert Pearcy (Yplant) and Belinda Medlyn (MAESTRA).

YuGene YuGene for comparing gene expression across platforms. Authors: Leo McHugh, Kim-Anh Le Cao.

ZeBook ZeBook Working with dynamic models for agriculture and environment. Authors: Francois Brun, David Makowski, Daniel Wallach, James W. Jones.

aBioMarVsuit A Biomarker Validation Suit for predicting Survival using gene signature. Authors: Pushpike Thalikarathne and Ziv Shkedy. In view: *Survival*.

aCRM Convenience functions for analytical Customer Relationship Management. Authors: Dirk Van den Poel, Michel Ballings, Andrey Volkov, Jeroen D'haen, Michiel Vanherwegen.

abctools Tools for ABC analyses. Authors: Matt Nunes and Dennis Prangle.

agrmt Calculate agreement. Author: Didier Ruedin.

akmeans Adaptive Kmeans algorithm based on threshold. Author: Jungsuk Kwac.

alr4 Data to accompany "Applied Linear Regression" 4rd edition. Author: Sanford Weisberg.

aods3 Analysis of Overdispersed Data using S3 methods. Authors: Matthieu Lesnoff and Renaud Lancelot.

arfima Fractional ARIMA Time Series Modeling. Authors: Justin Q. Veenstra, A.I. McLeod. In view: *TimeSeries*.

argparse Command line optional and positional argument parser. Author: Trevor L Davis. Ports examples from the argparse Python module by the Python Software Foundation. Ports examples from the **getopt** package by Allen Day.

astro Astronomy Functions, Tools and Routines. Author: Lee Kelvin. In view: *ChemPhys*.

audiolyzR Give your data a listen. Authors: Eric Stone, Jesse Garrison.

bPeaks A simple and intuitive approach for detection of basic peaks (bPeaks) from ChIP-seq data. Authors: Jawad MERHEJ and Gaelle LELANDAIS.

bagRboostR Ensemble bagging and boosting classifiers. Author: Shannon Rush.

bayess Bayesian Essentials with R. Authors: Christian P. Robert, Universite Paris Dauphine, and Jean-Michel Marin, Universite Montpellier 2.

bbo Biogeography-Based Optimization. Authors: Sarvesh Nikumbh (BBO originally invented by Prof. Dan Simon, Cleveland State University, Ohio).

bgeva Binary Generalized Extreme Value Additive Models. Authors: Raffaella Calabrese, Giampiero Marra and Silvia Anlgela Osmetti.

bigRR Generalized Ridge Regression (with special advantage for $p \gg n$ cases). Authors: Xia Shen, Moudud Alam and Lars Ronnegard.

bigrf Big Random Forests: Classification and Regression Forests for Large Data Sets. Authors: Aloysius Lim, Leo Breiman, Adele Cutler. In views: *HighPerformanceComputing*, *MachineLearning*.

biom Interface (beta) for the BIOM file format. Authors: Paul J. McMurdie and the biom-format team.

bmrm Bundle Methods for Regularized Risk Minimization Package. Author: Julien Prados.

bnpmr Bayesian monotonic nonparametric regression. Author: Bjoern Bornkamp.

boilerpipeR Interface to the boilerpipe Java library by Christian Kohlschutter (`http://code.google.com/p/boilerpipe/`). Author: Mario Annau [aut, cre].

bride Brier score decomposition of probabilistic forecasts for binary events. Author: Stefan Siegert.

brnn Bayesian regularization for feed-forward neural networks. Authors: Paulino Perez Rodriguez, Daniel Gianola.

bscr Bayesian parametric and semi-parametric analyses for semi-competing risks data. Authors: Kyu Ha Lee, Sebastien Haneuse, Deborah Schrag, and Francesca Dominici. In view: *Survival*.

c060 Additional variable selection, model validation and parameter tuning functions for glmnet models. Authors: Martin Sill, Thomas Hielscher, Manuela Zucknick, Natalia Becker.

cape Combined analysis of pleiotropy and epistasis. Authors: Anna L. Tyler, Wei Lu, Justin J. Hendrick, Vivek M. Philip, and Greg W. Carter.

catenary Fits a catenary to given points. Authors: Jonathan Tuke, Matthew Roughan.

causalsens Selection Bias Approach to Sensitivity Analysis for Causal Effects. Author: Matthew Blackwell.

cdb Reading and Writing Constant DataBases. Author: Emilio Torres Manzanera [aut, cre].

cec2013 Benchmark functions for the Special Session and Competition on Real-Parameter Single Objective Optimization at CEC-2013. Authors: Mauricio Zambrano-Bigiarini [aut, cre], Yasser Gonzalez Fernandez [aut].

celestial Collection of common astronomical conversion routines. Author: Aaron Robotham.

cgAUC Calculate AUC-type measure when gold standard is continuous and the corresponding optimal linear combination of variables with respect to it. Authors: Yuan-chin I. Chang, Yu-chia Chang, and Ling-wan Chen.

chebpol Multivariate Chebyshev interpolation. Author: Simen Gaure.

chillR Statistical methods for phenology analysis in temperate fruit trees. Author: Eike Luedeling.

citccmst CIT Colon Cancer Molecular SubTypes Prediction. Author: Laetitia Marisa.

cladoRcpp C++ implementations of phylogenetic calculations. Author: Nicholas J. Matzke [aut, cre, cph].

cldr Language Identifier based on CLD library. Authors: The Chromium Authors, Mike McCandless, Matt Sanford, Aykut Firat.

clinUtiDNA Clinical Utility of DNA Testing. Author: Thuy Trang Nguyen.

cocor Comparing correlations. Author: Birk Diedenhofen.

coefficientalpha Robust Cronbach's alpha with missing and non-normal data. Authors: Zhiyong Zhang and Ke-Hai Yuan.

colorfulVennPlot Plot and add custom coloring to Venn diagrams for 2-dimensional, 3-dimensional and 4-dimensional data. Authors: Elliot Noma, Aliona Manvae.

compositionsGUI Graphical User Environment for Compositional Data Analysis. Authors: Jiri Eichler, Karel Hron, Raimon Tolosana-Delgado, Gerald van den Boogaart, Matthias Templ, Peter Filzmoser.

conting Bayesian analysis of contingency tables. Author: Antony M. Overstall.

corHMM Analysis of binary character evolution. Authors: Jeremy M. Beaulieu, Jeffrey C. Oliver, Brian O'Meara.

covTest Computes covariance test for adaptive linear modelling. Authors: Richard Lockhart, Jon Taylor, Ryan Tibshirani, Rob Tibshirani.

currentSurvival Estimation of CCI and CLFS Functions. Authors: Eva Janousova, Tomas Pavlik, Jiri Mayer, Ladislav Dusek. In view: *Survival*.

cvAUC Cross-Validated Area Under the ROC Curve Confidence Intervals. Authors: Erin LeDell, Maya Petersen, Mark van der Laan.

cvxclustr Splitting methods for convex clustering. Author: Eric C. Chi.

cwm Cluster Weighted Models by EM algorithm. Authors: Giorgio A. Spedicato, Simona C. Minotti.

datacheck Tools for checking data consistency. Author: Reinhard Simon.

dataone DataONE R Client. Authors: Matthew Jones, Rob Nahf.

dataonelibs DataONE R Client Libraries. Authors: Matthew Jones, Rob Nahf.

datautils Support functions for packages **VBmix**, **semisupKernelPCA**, and **patchPlot**. Author: Pierrick Bruneau.

dave Functions for "Data Analysis in Vegetation Ecology". Author: Otto Wildi.

dbEmpLikeNorm Test for joint assessment of normality. Authors: Lori A. Shepherd, Wan-Min Tsai, Albert Vexler, Jeffrey C. Miecznikowski.

dclong.spt Sequential Permutation Test. Author: Chuanlong (Ben) Du.

ddalpha DDalpha-Classifier. Authors: Oleksii Pokotylo, Pavlo Mozharovskyi.

demi Differential Expression from Multiple Indicators. Authors: Sten Ilmjarv and Hendrik Luuk.

dendroextras Extra functions to cut, label and colour dendrogram clusters. Author: Gregory Jefferis.

designGG Computational tool for designing genetical genomics experiments. Authors: Yang Li, Morris Swertz, Gonzalo Vera, Rainer Breitling, Ritsert Jansen.

df2json Convert a dataframe to JSON. Author: Nacho Caballero.

diffdepprop Calculates Confidence Intervals for two Dependent Proportions. Authors: Daniela Wenzel, Antonia Zapf.

discreteRV Author: Andreas Buja.

discrimARTs Discrimination of Alternative Reproductive Tactics (ARTs). Authors: J. Mark Rowland, Clifford Qualls, and Christian Gunning.

displayHTS Author: Xiaohua Douglas Zhang & Zhaozhi Zhang. In view: *ExperimentalDesign*.

distory Distance Between Phylogenetic Histories. Authors: John Chakerian and Susan Holmes. In view: *Phylogenetics*.

distrRmetrics Package **distr** classes for distributions from Rmetrics. Authors: Nataliya Horbenko, Matthias Kohl, Daria Pupashenko, Myhailo Pupashenko, Peter Ruckdeschel.

driftsel Distinguishing drift and natural selection as causes of phenotypic differentiation. Author: Markku Karhunen & Otso Ovaskainen.

dvn Access to The Dataverse Network API. Author: Thomas J. Leeper.

eHOF Extended and enhanced Hierarchical Logistic Regression models (so called Huisman-Olff-Fresco models). Authors: Florian Jansen, Jari Oksanen.

earlywarnings Early Warning Signals Toolbox for Detecting Critical Transitions in Time-series. Authors: Vasilis Dakos, with contributions from S.R. Carpenter, T. Cline, L. Lahti.

easystab Clustering Perturbation Stability Analysis. Authors: Hoyt Koepke, Zongjun Hu, Bertrand Clarke.

ecosim Toolbox for Aquatic Ecosystem Modeling. Author: Peter Reichert.

ecp Nonparametric Multiple Change Point Analysis of Multivariate Data. Authors: Nicholas A. James and David S. Matteson.

eive An algorithm for reducing errors-in-variable bias in simple linear regression. Authors: Mehmet Hakan Satman, Erkin Diyarbakirlioglu.

ensembleMOS Ensemble Model Output Statistics. Authors: RA Yuen, Tilmann Gneiting, Thordis Thorarinsdottir, Chris Fraley.

epibase basic tools for the analysis of disease outbreaks. Authors: The Hackout team (In alphabetic order: David Aanensen, Marc Baguelin, Paul Birrell, Simon Cauchemez, Anton Camacho, Caroline Colijn, Anne Cori, Xavier Didelot, Ken Eames, Christophe Fraser, Simon Frost, Niel Hens, Joseph Hugues, Thibaut Jombart, Lulla Opatowski, Oliver Ratmann, Samuel Soubeyrand, Marc Suchard, Jacco Wallinga, Rolf Ypma).

ergm.count Fit, Simulate and Diagnose Exponential-Family Models for Networks with Count Edges. Authors: Pavel N. Krivitsky [aut, cre], Mark S. Handcock [ctb], David R. Hunter [ctb].

ergmharris Local Health Department network data set. Author: Jenine K. Harris.

eventstudies Event study and extreme event analysis. Authors: Ajay Shah, Vimal Balasubramaniam, Vikram Bahure.

expoTree Calculate density dependent likelihood of a phylogenetic tree. Authors: Gabriel E Leventhal, partly adapted from MATLAB code by Awad H. Al-Mohy and using the routines DLNAC1 and DLARPC by Sheung Hun Cheng, and DLAPST from ScaLAPACK.

exsic Convenience functions for botanist to create exsiccatae indices. Authors: Reinhard Simon, David M. Spooner.

fail File Abstraction Interface Layer (FAIL) mimicking a key-value store. Author: Michel Lang.

farsi Translate integers into persian. Author: Sadegh Rasoulinejad.

fastclime A fast solver for constrained l1 minimization approach to sparse precision matrix estimation. Authors: Haotian Pang, Han Liu and Robert Vanderbei.

fbRanks Association Football (Soccer) Ranking via Poisson Regression. Author: Eli Holmes.

fftwtools Author: Karim Rahim.

fit.models Author: Kjell Konis.

gIPFrm Generalized Iterative Proportional Fitting for Relational Models. Authors: Anna Klimova, Tamas Rudas.

gains Gains Table Package. Author: Craig A. Rolling.

gamlr Gamma Lasso Regression. Author: Matt Taddy.

gaoptim Genetic Algorithm optimization for real-based and permutation-based problems. Author: Fernando Tenorio.

gazetools Gaze Tools. Author: Ryan M. Hope.

gcookbook Data for "R Graphics Cookbook". Author: Winston Chang.

geigen Calculate generalized eigenvalues of a matrix pair. Author: Berend Hasselman.

gems Generalized multistate simulation model. Authors: Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer. In view: *Survival*.

genMOSS An implementation of the MOSS algorithm for the analysis of GWAS data. Authors: Matthew Friedlander and Laurent Briollais.

gensemble Generalized ensemble methods. Authors: Peter Werner, Eugene Dubossarsky.

geoscale Geological timescale plot. Author: Mark A. Bell.

geospacom Helper package to facilitate the generation of distance matrices used in the package **spacom**. Authors: Davide Morselli [aut], Mathieu Cossuta [aut, cre], Till Junge [aut], Sandra Penic [aut], Guy Elcheroth [ctb], Stephanie Glaeser [ctb].

ggdendro Tools for extracting dendrogram and tree diagram plot data for use with **ggplot**. Authors: Andrie de Vries, Brian Ripley.

ggthemes Extra themes, scales and geoms for **ggplot**. Author: Jeffrey B. Arnold.

glassomix High dimensional Sparse Gaussian Graphical Mixture Model. Authors: Anani Lotsi and Ernst Wit.

gplm Generalized partial linear models (GPLM). Author: Marlene Mueller.

grnn General regression neural network. Author: Pierre-Olivier Chasset.

gsg Calculation of selection coefficients. Authors: Michael Morrissey, Krzysztof Sakrejda.

gte Generalized Turnbull's estimator. Authors: Mohammad Hossein Dehghan, Thierry Duchesne and Sophie Baillargeon.

hSDM Hierarchical Bayesian species distribution models. Authors: Ghislain Vieilledent, Andrew M. Latimer, Alan E. Gelfand, Cory Merow, Adam M. Wilson, Frederic Mortier and John A. Silander Jr.

hashFunction A collection of non-cryptographic hash functions. Author: Xiaowei Zhan.

hcc Hidden correlation check. Authors: Yun Shi and A. I. McLeod.

heatex Heat exchange calculations during physical activity. Author: Kerry Atkins.

het.test White's Test for Heteroskedasticity. Author: Sebastian Andersson.

hht The Hilbert-Huang Transform: Tools and Methods. Author: Daniel Bowman [aut, cre].

highfrequency Authors: Jonathan Cornelissen, Kris Boudt, Scott Payseur. In view: *Finance*.

highriskzone Determining and evaluating high-risk zones. Authors: Heidi Seibold, Monia Mahling.

hint Tools for hypothesis testing based on Hypergeometric Intersection distributions. Author: Alex T. Kalinka.

hsicCCA Canonical Correlation Analysis based on Kernel Independence Measures. Author: Billy Chang.

httpuv HTTP and WebSocket server library. Authors: RStudio, Inc.

iWeigReg Improved methods for causal inference and missing data problems. Authors: Zhiqiang Tan and Heng Shu.

ibd Incomplete Block Designs. Author: Baidya Nath Mandal.

inflection Finds the inflection point of a curve. Author: Demetris T. Christopoulos.

installr Functions for updating and installing a new version of R and other software from R. Author: Tal Galili.

jaatha A Fast Parameter Estimation Method for Evolutionary Biology. Authors: Lisha Mathew, Paul R. Staab and Dirk Metzler.

jackknifeKME Jackknife estimates of Kaplan-Meier estimators or integrals. Authors: Hasinur Rahaman Khan and Ewart Shaw. In view: *Survival*.

kdetrees Nonparametric method for identifying discordant trees. Authors: Grady Weyenberg and Peter Huggins.

kernelFactory Kernel Factory: An ensemble of kernel Machines. Authors: Michel Ballings, Dirk Van den Poel.

kolmim An improved evaluation of Kolmogorov's distribution. Author: Luis Carvalho.

lazyData A LazyData Facility. Author: Bill Venables.

lbiassurv Length-biased correction to survival curve estimation. Authors: Pierre-Jerome Bergeron and Vahid Partovi Nia. In view: *Survival*.

lfstat Calculates Low Flow Statistics for daily stream flow data. Author: Daniel Koffler.

lisrelToR Import output from LISREL into R. Author: Sacha Epskamp.

llama Leveraging Learning to Automatically Manage Algorithms. Authors: Lars Kotthoff, contributions by Barry Hurley.

lmeNB Fit negative binomial mixed-effect regression model. Authors: Zhao, Y. and Kondo, Y.

lmeNBBayes Sample from the posterior of the negative binomial mixed-effect regression model. The random effect is modeled with DP mixture of beta distributions. Author: Yumi Kondo.

lmerTest Tests for random and fixed effects for linear mixed effect models (lmer objects of **lme4** package). Authors: Alexandra Kuznetsova, Per Bruun Brockhoff, Rune Haubo Bojesen Christensen.

localgauss Estimating local Gaussian parameters. Author: Tore Selland Kleppe.

logistiX Exact logistic regression including Firth correction. Authors: Georg Heinze and Tobias Ladner.

longCatEDA Plotting Categorical Longitudinal and Time-Series Data. Author: Stephen Tueller.

ltmle Longitudinal Targeted Maximum Likelihood Estimation. Authors: Joshua Schwab, Maya Petersen, and Mark van der Laan, with contributions from Susan Gruber.

madsim A Flexible Microarray Data Simulation Model. Author: Doulaye Dembele.

margie Log Marginal Likelihood of Gaussian Mixture. Authors: Elisa Loza [aut, cre], David Phillips [aut, com].

marked R Code for mark-recapture analysis. Authors: Jeff Laake, Devin Johnson, Paul Conn.

marmap Import, plot and analyze bathymetric and topographic data. Authors: Eric Pante and Benoit Simon-Bouhet.

matie Measuring Association and Testing Independence Efficiently. Authors: Ben Murrell, Dan Murrell & Hugh Murrell.

mcIRT IRT models for multiple choice items (mcIRT). Author: Manuel Reif. In view: *Psychometrics*.

mchof multicore higher-order functions. Author: Ryan Grannell. In view: *HighPerformanceComputing*.

medSTC A max-margin supervised Sparse Topical Coding Model. Authors: Jun Zhu, Aykut FIRAT.

melody Statistical Methods for the Quantitative Analysis of Song Spectrograms. Author: Dave Schruth.

metabolomics A collection of functions for analysing metabolomics data. Authors: Alysha M De Livera and Jairus B Bowne.

metacom Analysis of the 'elements of metacommunity structure'. Author: Tad Dallas. In view: *Environmetrics*.

metagen Generalised Inference in the Random Effects Meta Regression Model. Author: Thomas Friedrich. In view: *MetaAnalysis*.

meteogRam Tools for plotting meteograms. Author: Bogdan Bochenek.

micromap Linked Micromap Plots. Authors: Quinn Payton and Tony Olsen with contributions from Marc Weber, Michael McManus, and Tom Kincaid. In view: *Spatial*.

miscF Miscellaneous Functions. Author: Dai Feng.

mistral Methods in Structural Reliability. Authors: Vincent Moutoussamy, Nicolas Bousquet and Bertrand Iooss.

mixture Mixture Models for Clustering and Classification. Authors: Ryan P. Browne and Paul D. McNicholas.

mlearning Machine learning algorithms with unified interface and confusion matrices. Author: Ph. Grosjean & K. Denis.

modiscloud R tools for processing Level 2 Cloud Mask products from MODIS. Authors: Nicholas J. Matzke, Dept. of Integrative Biology, U.C. Berkeley.

move Visualizing and analyzing animal track data. Authors: Bart Kranstauber, Marco Smolla.

msarc Draws diagrams (mis)representing the results of mass spec experiments. Authors: Gord Brown, Hisham Mohammed.

msgl High dimensional multiclass classification using sparse group lasso. Authors: Martin Vincent (Sparse group lasso), Conrad Sanderson (Armadillo), Jaakko Jarvi (Boost Tuple) and Jens Maurer (Boost Random).

msgpackR A library to serialize or unserialize data in MessagePack format. Author: Mikiya Tanizawa.

msme Functions and Datasets for "Methods of Statistical Model Estimation". Authors: Joseph Hilbe and Andrew Robinson.

multilevelPSA Multilevel Propensity Score Analysis. Author: Jason Bryer.

mvinfluence Influence Measures and Diagnostic Plots for Multivariate Linear Models. Author: Michael Friendly.

nCDunnett Noncentral Dunnett's test distribution. Authors: Siomara Cristina Broch, Daniel Furtado Ferreira.

nCal Nonlinear Calibration. Authors: Youyi Fong, Krisztian Sebestyen.

ncbit Retrieve and build NBCI taxonomic data. Author: Jon Eastman.

nephro Biostatistics utilities for nephrology. Author: Cristian Pattaro.

nets Network Estimation for Time Series. Author: Christian Brownlees. In view: *TimeSeries*.

networkTomography Tools for network tomography. Authors: Alexander W Blocker, Paul Koullick, Edoardo Airoldi.

neuroblastoma Author: Toby Dylan Hocking.

nloptwrap Wrapper for Package **nloptr**. Author: Hans W Borchers.

nordklimdata1 Dataset for climate analysis with data from the Nordic region. Authors: Heikki Tuomenvirta [aut], Achim Drebs [aut], Eirik Forland [aut], Ole Einar Tveito [aut], Hans Alexandersson [aut], Ellen Vaarby Laursen [aut], Trausti Jonsson [aut], Jose' Gama [cre].

nose Author: originally written by Subbiah M, packaged by Sumathi R with considerable contributions by Srinivasan M R.

noweb Noweb system for R. Author: Terry Therneau.

nsprcomp Non-Negative Sparse PCA. Authors: Sigg Christian [aut, cre], R Core team [aut].

nutshell.audioscrobbler Audioscrobbler data for "R in a Nutshell". Author: Joseph Adler.

nutshell.bbdb Baseball Database for "R in a Nutshell". Author: Joseph Adler.

optAUC Optimal Combinations of Diagnostic Tests Based on AUC. Authors: Xin Huang, Gengsheng Qin, Yixin Fang.

outbreaker Bayesian inference of outbreak dynamics based on epidemiological and genetic information. Authors: Thibaut Jombart, Anne Cori, Xavier Didelot, Simon Cauchemez, Christophe Fraser, Neil Ferguson.

overlap Estimates of coefficient of overlapping for animal activity patterns. Authors: Mike Meredith and Martin Ridout.

packHV A few useful functions for statisticians. Author: Hugo Varet.

pairwise Rasch Model Parameters by Pairwise Algorithm. Author: Joerg-Henrik Heine.

parallelize.dynamic Automate parallelization of function calls by means of dynamic code analysis. Author: Stefan Boehringer.

parma Portfolio Allocation and Risk Management Applications. Author: Alexios Ghalanos. In view: *Finance*.

partDSA Partitioning using deletion, substitution, and addition moves. Authors: Annette Molinaro, Karen Lostritto, Gregory Ryslik, Steve Weston. In view: *HighPerformanceComputing*.

partialOR Partial Odds Ratio. Authors: Vaclav Fidler and Nico Nagelkerke.

pass Prediction and Stability Selection of Tuning Parameters. Authors: Yixin Fang, Wei Sun, Junhui Wang.

patchDVI Patch '.dvi' files. Author: Duncan Murdoch.

patchPlot Scatterplots of image patches. Author: Pierrick Bruneau.

pavo Perceptual analysis, visualization and organization of spectral color data in R. Authors: Rafael Maia [aut, cre], Chad Eliason [aut], Pierre-Paul Bitton [aut].

pbdDEMO Programming with Big Data – Demonstrations of pbd Packages. Authors: Drew Schmidt [aut, cre], Wei-Chen Chen [aut], George Ostrouchov [aut], Pragneshkumar Patel [aut]. In view: *HighPerformanceComputing*.

pbdNCDF4 Programming with Big Data – Interface to Parallel Unidata NetCDF4 Format Data Files. Authors: Pragneshkumar Patel [aut, cre], George Ostrouchov [aut], Wei-Chen Chen [aut], Drew Schmidt [aut], David Pierce [aut]. In views: *HighPerformanceComputing*, *SpatioTemporal*.

pca3d Three dimensional PCA plots. Author: January Weiner.

pcrsim Simulation of the forensic DNA process. Author: Oskar Hansson.

pearson7 Maximum Likelihood Inference for the Pearson VII Distribution with Shape Parameter 3/2. Author: John Hughes.

perARMA Periodic Time Series Analysis. Authors: Anna Dudek, Harry Hurd and Wioletta Wojtowicz. In view: *TimeSeries*.

permGPU Using GPUs in Statistical Genomics. Authors: Ivo D. Shterev, Sin-Ho Jung, Stephen L. George and Kouros Owzar. In view: *HighPerformanceComputing*.

persiandictionary English to Persian dictionary. Authors: Sadegh Rasoulinejad (R package and database version of the dictionary) Manouchehr Aryanpour (for the original dictionary).

pfa Estimates False Discovery Proportion Under Arbitrary Covariance Dependence. Authors: Jianqing Fan, Tracy Ke, Sydney Li and Lucy Xia.

phenex Auxiliary functions for phenological data analysis. Authors: Lange, Maximilian and Doktor, Daniel.

phtt Panel Data Analysis with Heterogeneous Time Trends. Authors: Oualid Bada, Dominik Liebl.

phylolm Phylogenetic Linear Regression. Authors: Lam Si Tung Ho, Cecile Ane.

pitchRx Tools for Collecting and Visualizing Major League Baseball PITCHf/x Data. Author: Carson Sievert.

plfm Probabilistic latent feature analysis of two-way two-mode frequency data. Author: Michel Meulders.

pmmlTransformations Transform input data from a PMML perspective. Author: Tridivesh Jena.

pnn Probabilistic neural networks. Author: Pierre-Olivier Chasset.

pocrm Dose-finding in drug combination Phase I trials using the partial order continual reassessment method (PO-CRM). Author: Nolan A. Wages.

polyCub Cubature over Polygonal Domains. Authors: Sebastian Meyer [aut, cre, trl], Michael Hoehle [ths].

pom Patch Occupancy Models. Authors: Fawn Hornsby, Ryan Nielson, and Trent McDonald (www.west-inc.com).

popgraph Construct and manipulate population graphs. Author: Rodney J. Dyer.

poppr Genetic analysis of populations with mixed reproduction. Authors: Zhian N. Kamvar, Javier F. Tabima, Niklaus J. Grunwald.

probsvm Class probability estimation for Support Vector Machines. Authors: Chong Zhang, Seung Jun Shin, Junhui Wang, Yichao Wu, Hao Helen Zhang, and Yufeng Liu.

proteomicdesign Optimization of a multi-stage proteomic study. Author: Irene SL Zeng.

psbcGroup Penalized semi-parametric Bayesian Cox (PSBC) models with shrinkage and grouping priors. Authors: Kyu Ha Lee, Sounak Chakraborty, (Tony) Jianguo Sun.

psd Adaptive, sine-multitaper power spectral density estimation. Authors: Robert L. Parker and Andrew J. Barbour.

qdap Bridging the gap between qualitative data and quantitative analysis. Author: Tyler Rinker. In view: *NaturalLanguageProcessing*.

quantregGrowth Growth charts via regression quantiles. Author: Vito M. R. Muggeo.

questionr Functions to make surveys processing easier. Authors: Julien Barnier, François Briatte.

rCMA CMA-ES R-to-Java interface. Author: Wolfgang Konen.

rCarto Build maps with a full cartographic layout. Author: Timothee Giraud.

rGammaGamma Gamma convolutions for methylation array background correction. Author: Tim Triche, Jr.

rHadoopClient hadoop client interface for R. Author: Yohei Sato.

rPython Allows R to call Python. Author: Carlos J. Gil Bellosta.

randomGLM Random General Linear Model Prediction. Authors: Lin Song, Peter Langfelder.

rattle Graphical user interface for data mining in R. Authors: Graham Williams [aut, cph, cre], Mark Vere Culp [cph], Ed Cox [ctb], Anthony Nolan [ctb], Denis White [cph], Daniele Medri [ctb], Akbar Waljee [ctb] (OOB AUC for Random Forest). In view: *MachineLearning*.

rawFasta Memory efficient handling of FASTA sequence files, a pure R implementation. Author: Sylvain Mareschal.

rbefdata BEFdata R package. Authors: Claas-Thido Pfaff, Karin Nadrowski.

readMETEO Weather Data Download from www.meteogalicia.com. Author: Dominic Roye.

recluster Solving biasing produced by tied values in cluster analysis. Authors: Leonardo Dapporto, Matteo Ramazzotti, Simone Fattorini, Roger Vila, Gerard Talavera, Roger H.L. Dennis.

refGenome Managing data from reference genomes (UCSC, ensembl). Author: Wolfgang Kaisers.

relaxnet Relaxation of glmnet models (as in relaxed lasso, Meinshausen 2007). Authors: Stephan Ritter, Alan Hubbard.

repmis A collection of miscellaneous tools for reproducible research with R. Author: Christopher Gandrud.

reports Assist in the workflow of writing academic articles and other reports. Author: Tyler Rinker.

restorepoint Debugging with restore points. Author: Sebastian Kranz.

rexpokit R wrappers for EXPOKIT; other matrix functions. Authors: Nicholas J. Matzke [aut, cre, cph], Roger B. Sidje [aut, cph].

rforensicbatwing BATWING for calculating forensic trace-suspect match probabilities. Authors: Mikkel Meyer Andersen and Ian J. Wilson.

ringbuffer Ring buffer (circular buffer) data structure. Author: Allen Day.

rlme Random Effects Nested Models. Authors: Yusuf Bilgic and Herb Susmann.

rmgarch Multivariate GARCH models. Author: Alexios Ghalanos. In view: *Finance*.

rmp Rounded Mixture Package. Performs probability mass function estimation with Dirichlet process mixtures of rounded kernels. Authors: C code by A. Canale and N. Lunardon, port and R code by A. Canale.

rms.gof Root-mean-square goodness-of-fit test for simple null hypothesis. Author: Shubhodeep Mukherji.

robustlmm Robust Linear Mixed Effects Models. Author: Manuel Koller. In view: *Robust*.

roxyPackage Utilities to automate package builds. Author: m.eik michalke.

royston Royston's H Test: Multivariate Normality Test. Author: Selcuk Korkmaz.

rspear Calculate SPEARpesticide in R (`http://www.systemecology.eu/SPEAR/index.php`). Author: Eduard Szoecs.

rtop Interpolation of data with variable spatial support. Author: Jon Olav Skoien.

sae Small Area Estimation. Authors: Isabel Molina, Yolanda Marhuenda.

sanon Stratified Analysis with Nonparametric covariable adjustment. Author: Atsushi Kawaguchi.

schoRsch Tools for analyzing factorial experiments. Authors: Roland Pfister, Markus Janczyk.

scidb An R interface to SciDB. Authors: Paradigm4, B. W. Lewis.

seeg Statistics for Environmental Sciences, Engineering, and Geography. Author: Miguel F. Acevedo.

semPlot Path diagrams and visual analysis of various SEM packages' output. Author: Sacha Epskamp.

semisupKernelPCA Kernel PCA projection, and semi-supervised variant. Author: Pierrick Bruneau.

seqPERM Generates a permutation matrix based upon a sequence. Author: Eric Golinko.

seqminer Efficiently Read Sequencing Data (VCF format, METAL format) into R. Authors: Xiaowei Zhan and Dajiang Liu, with contributions of Jean-loup Gailly and Mark Adler (zlib), Julian Seward (bzip2) and Heng Li (tabix).

sequences Generic and biological sequences. Authors: Laurent Gatto and Robert Stojnic.

severity Mayo's Post-data Severity Evaluation. Author: Nicole Mee-Hyaang Jinn.

sgof Multiple hypotheses testing. Authors: Irene Castro Conde and Jacobo de Una Alvarez.

sgr Sample Generation by Replacement. Author: Massimiliano Pastore & Luigi Lombardi.

shiny Web Application Framework for R. Authors: RStudio, Inc.

shrink Global, Parameterwise, and Joint Shrinkage of Regression Coefficients. Authors: Daniela Dunkler, Georg Heinze.

sidier Substitution and Indel Distances to Infer Evolutionary Relationships. Author: A.J. Munoz-Pajares.

skatMeta Efficient meta analysis for the SKAT test. Authors: Arend Voorman, Jennifer Brody, Thomas Lumley. In view: *MetaAnalysis*.

sltl Time Series Decomposition using Loess and Harmonic Regression. Authors: Hyukjun Gweon and A.I. McLeod.

smdata Data to accompany Smithson & Merkle, 2014. Authors: Ed Merkle and Michael Smithson.

smdc Document Similarity. Author: Masaaki Takada.

smoothHR Smooth Hazard Ratio Curves taking a Reference Value. Authors: Artur Agostinho Araujo and Luis Meira-Machado.

snpEnrichment SNPs enrichment analysis. Authors: Mickael Canouil [aut, cre], Loic Yengo [ctb].

snpStatsWriter Flexible writing of snpStats objects to flat files. Author: Chris Wallace.

softImpute Matrix completion via iterative soft-thresholded SVD. Authors: Trevor Hastie and Rahul Mazumder.

source.gist Read R code from a GitHub Gist. Author: Mason Simon.

spacejam Sparse conditional graph estimation with joint additive models. Author: Arend Voorman.

sparsediscrim Sparse Discriminant Analysis. Author: John A. Ramey.

spate Spatio-temporal modeling of large data using a spectral SPDE approach. Authors: Fabio Sigrist, Hans R. Kuensch, Werner A. Stahel. In view: *SpatioTemporal*.

spatial.tools R functions for working with spatial data. Author: Jonathan Asher Greenberg.

specificity Specificity of personality trait-outcome associations. Authors: Kenn Konstabel, Rene Mottus.

speedRlibTF speedR's table filter library. Author: Ilhami Visne.

sprsmdl Sparse modeling toolkit. Author: Hiroshi Saito.

sqlshare API for access to SQLShare database. Author: Andrew White.

sqlutils Utilities for working with SQL files. Author: Jason Bryer.

stacomirtools stacomi ODBC connection class. Author: Cedric Briand.

statnet.common Common R Scripts and Utilities Used by the Statnet Project Software. Author: Pavel N. Krivitsky [aut, cre].

stochvol Efficient Bayesian Inference for Stochastic Volatility (SV) Models. Author: Gregor Kastner. In views: *Bayesian*, *Finance*, *TimeSeries*.

stream Infrastructure for Data Stream Mining. Authors: Matthew Bolanos, John Forrest, Michael Hahsler.

streamR Access to Twitter Streaming API via R. Author: Pablo Barbera.

stringdist String distance functions for R. Author: Mark van der Loo.

structSSI Multiple Testing for Hypotheses with Hierarchical or Group Structure. Author: Kris Sankaran.

strvalidator Internal validation of forensic STR kits made easy with strvalidator. Author: Oskar Hansson.

subtype Cluster analysis to find molecular subtypes and their assessment. Authors: Andrey Alexeyenko, Woojoo Lee and Yudi Pawitan.

superbiclust Generating Robust Biclusters from a Bicluster Set (Ensemble Biclustering). Author: Tatsiana Khamiakova.

support.BWS Basic functions for supporting an implementation of best-worst scaling. Author: Hideo Aizaki.

surface Fitting Hansen Models to Investigate Convergent Evolution. Author: Travis Ingram.

survMisc Miscellaneous functions for survival data. Author: Chris Dardis.

survexp.fr Relative survival, AER and SMR based on French death rates. Authors: Jean-Philippe Jais and Hugo Varet. In view: *Survival*.

surveydata Tools to manipulate survey data. Author: Andrie de Vries.

svDialogstcltk SciViews GUI API — Dialog boxes using Tcl/Tk. Author: Philippe Grosjean.

tagcloud Tag Clouds. Author: January Weiner.

taxize Taxonomic information from around the web. Authors: Scott Chamberlain [aut, cre], Eduard Szoecs [aut], Carl Boettiger [aut], Ignasi Bartomeus [aut].

tergm Fit, Simulate and Diagnose Models for Network Evoluation based on Exponential-Family Random Graph Models. Authors: Pavel N. Krivitsky [aut, cre], Mark S. Handcock [aut, ths], David R. Hunter [ctb], Steven M. Goodreau [ctb, ths], Martina Morris [ctb, ths], Nicole Bohme Carnegie [ctb], Carter T. Butts [ctb], Ayn Leslie-Cook [ctb].

threg Threshold Regression. Author: Tao Xiao.

timeit Easy profiling of R functions. Author: Kevin Ushey.

timeline Timelines for a Grammar of Graphics. Author: Jason Bryer.

tipom Automated measure-based classification for lithic tools. Authors: Stefano Costa [aut, cre], Luca Bianconi [aut], Elisabetta Starnini [ctb].

tm.plugin.webmining Retrieve structured, textual data from various web sources. Author: Mario Annau [aut, cre].

trioGxE A data smoothing approach to explore and test gene-environment interaction in case-parent trio data. Authors: Ji-Hyung Shin, Brad McNeney, Jinko Graham.

trueskill Implementation the TrueSkill algorithm in R. Author: Brendan Houng.

tsbugs Create time series BUGS models. Author: Guy J. Abel. In view: *TimeSeries*.

tspmeta Instance feature calculation and evolutionary instance generation for the traveling salesman problem. Authors: Bernd Bischl, Jakob Bossek, Olaf Mersmann.

tth TeX to HTML/MathML Translators tth/ttm. Authors: Ian H. Hutchinson [aut] (author of tth/ttm C sources), Friedrich Leisch [aut, cre] (author of R wrappers to tth/ttm), Achim Zeileis [aut] (author of R wrappers to tth/ttm).

twostageTE Two-Stage Threshold Estimation. Authors: Shawn Mankad, George Michailidis, Moulinath Banerjee.

usl Analyze system scalability with the Universal Scalability Law. Authors: Neil J. Gunther [aut], Stefan Moeding [aut, cre].

visualize Graph Probability Distributions with User Supplied Parameters and Stats. Author: James Balamuta.

vitality vitality: Fitting routines for the Vitality family of mortality models. Authors: Gregor Passolt, James J. Anderson, Ting Li, David H. Salinger.

wethepeople An R client for interacting with the White House's "We The People" petition API. Author: Yoni Ben-Meshulam.

witness Fits eyewitness data using Clark's (2003) WITNESS model. Author: Dustin Fife.

wpp2008 World Population Prospects 2008. Authors: Hana Sevcikova, Patrick Gerland.

wpp2010 World Population Prospects 2010. Authors: Hana Sevcikova, Patrick Gerland.

xkcd Plotting ggplot2 graphics in a XKCD style. Author: Emilio Torres Manzanera.

yhatr R binder for the Yhat API. Author: Greg Lamp.

zooimage Analysis of numerical zooplankton images. Authors: Ph. Grosjean, K. Denis & R. Francois.

Other changes

- The following packages were moved to the Archive: **AGSDest**, **AMA**, **BADER**, **BAMD**, **BAYSTAR**, **BPHO**, **BootPR**, **CADStat**, **CAVIAR**, **CarbonEL**, **CompModSA**, **ExPD2D**, **GAMens**, **GDD**, **ImageMetrics**, **LDdiag**, **LN3GV**, **MAMSE**, **MISA**, **MMS**, **MSToolkit**, **McSpatial**, **Metabonomic**, **MethComp**, **MixMod**, **NADA**, **NMMAPSlite**, **NeMo**, **PHYLOGR**, **PSCN**, **PrecipStat**, **QTLNetworkR**, **R4dfp**, **RDF**, **RMediation**, **RTAQ**, **RandForestGUI**, **Rchemcpp**, **Rclusterpp**, **RcmdrPlugin.Export**, **RcmdrPlugin.FactoMineR**, **RcmdrPlugin.MAc**, **RcmdrPlugin.MAd**, **RcmdrPlugin.PT**, **RcmdrPlugin.TextMining**, **RcmdrPlugin.depthTools**, **RcmdrPlugin.pointG**, **RcmdrPlugin.sos**, **RcmdrPlugin.steepness**, **ReadImages**, **Rigroup**, **Rlof**, **RpgSQL**, **Rsge**, **Sim.DiffProc**, **Sim.DiffProcGUI**, **Simile**, **SpectralGEM**, **StatFingerprints**, **TSpadi**, **TreePar**, **VBMA4hmm**, **WMBrukerParser**, **XLConnectJars**, **YourCast**, **anchors**, **aratio**, **ares**, **assist**, **bayespack**, **betfairly**, **bise**, **biseVec**, **boolean**, **bujar**, **cMonkey**, **calibFit**, **canvas**, **caspar**, **ccems**, **cems**, **clim.pact**, **cloudRmpi**, **cloudRmpiJars**, **cmprskContin**, **cond**, **csampling**, **cyclones**, **dataframe**, **delftfews**, **delt**, **denpro**, **diff**, **digeR**, **dynGraph**, **dynamo**, **emu**, **emudata**, **epsi**, **esd4all**, **fairselect**, **fdim**, **filterviewR**, **finebalance**, **fractal**, **gcolor**, **gearman**, **genomatic**, **graphComp**, **hdf5**, **helpr**, **highlight**, **iDEMO**, **iGenomicViewer**, **iid.test**, **integrativeME**, **kBestShortestPaths**, **laser**, **lazy**, **lmeSplines**, **marg**, **maxLinear**, **mcpd**, **medAdherence**, **mht**, **mixer**, **mmeta**, **mnspc**, **motmot**, **mpc**, **mprobit**, **mspath**, **nFDR**, **nlreg**, **nltm**, **nlts**, **npst**, **odesolve**, **paloma**, **pamctdp**, **parser**, **pcurve**, **pfda**, **phitest**, **phull**, **pmc**, **polydect**, **predbayescor**, **proptest**, **qqplotter**, **rPorta**, **rconifers**, **realized**, **replicationDemos**, **ritis**, **rpartOrdinal**, **rqmcmb2**, **rreval**, **rv**, **rvmbinary**, **s3x**, **sampfling**, **sculpt3d**, **sentiment**, **simone**, **sound**, **speedR**, **speedRlibTF**, **splinesurv**, **sspline**, **steepness**, **sudokuplus**, **surv2sample**, **tabplotGTK**, **taskPR**, **tsModel**, **two.stage.boot**, **unknownR**, **xlsReadWrite**

- The following packages were resurrected from the Archive: **ArDec**, **Bhat**, **Biograph**, **CONOR**, **DCluster**, **EMT**, **ForImp**, **GWAtoolbox**, **MFDA**, **NMF**, **PairedData**, **SpatialEpi**, **bdoc**, **brainwaver**, **bspec**, **bwsurvival**, **covRobust**, **crawl**, **crossdes**, **cwhmisc**, **cyphid**, **formula.tools**, **fptdApprox**, **fso**, **hsmm**, **iteRates**, **labeltodendro**, **maticce**, **networksis**, **nricens**, **onemap**, **operator.tools**, **optpart**, **orderbook**, **partitionMap**, **pass**, **seas**, **snowfall**, **support.BWS**, **surveillance**, **tipom**, **trueskill**, **truncgof**

- The following packages had to be removed: **RExcelInstaller**, **RthroughExcelWorkbooksInstaller**, **SWordInstaller**, **auteur**, **rcom**, **wvioplot**

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Achim Zeileis
Universität Innsbruck, Austria
Achim.Zeileis@R-project.org